
Pós-Graduação em Ciência da Computação

Thiago Vinicius Machado de Souza

# Utilizando Spatial Transformer Networks no agrupamento de imagens baseado em Deep Adaptive Clustering

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2019

Thiago Vinicius Machado de Souza

# Utilizando Spatial Transformer Networks no agrupamento de imagens baseado em Deep Adaptive Clustering

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

**Área de Concentração**: inteligência computacional
**Orientador**: Cleber Zanchettin

Recife
2019

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S729u Souza, Thiago Vinicius Machado de
Utilizando *spatial transformer networks* no agrupamento de imagens baseado em *deep adaptive clustering* / Thiago Vinicius Machado de Souza. – 2019.
71 f.: il., fig., tab.

Orientador: Cleber Zanchettin.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2019.
Inclui referências e apêndices.

1. Inteligência computacional. 2. Agrupamento de imagens. I. Zanchettin, Cleber (orientador). II. Título.

006.3 CDD (23. ed.) UFPE- MEI 2019-064

Dissertação de Mestrado apresentada por **Thiago Vinicius Machado de Souza** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Utilizando Spatial Transformer Networks no agrupamento de imagens baseado em Deep Adaptive Clustering**" **Orientador: Cleber Zanchettin** e aprovada pela Banca Examinadora formada pelos professores:

Aprovado em: 26/02/2019

---

Prof. Dr. Carlos Alexandre Barros de Mello
Centro de Informática/ UFPE

---

Prof. Dr. Bruno José Torres Fernandes
Escola Politécnica de Pernambuco - POLI / UPE

---

Prof. Dr. **Orientador**: Cleber Zanchettin
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 26 de Fevereiro de 2019.

---

**Prof. Ricardo Bastos Cavalcante Prudêncio**
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

*Decido este trabalho a minha família e minha namorada que foram porto seguro perante as dificuldades durante este percurso.*

## AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter me dado saúde e perseverança para realizar esse objetivo. Agradeço à minha Família, especialmente aos meus pais, Zenilda Machado de Souza e José Ferreira de Souza, a minha irmã Barbára Licia Machado de Souza, e a minha namorada Ruth Rodrigues, por terem me apoiado em todos os momentos e me possibilitado momentos de felicidades. Agradeço ao meu orientador Prof. Cleber Zanchettin e ao meu amigo de trabalho e vida Everton Lacerda, por toda a paciência, disponibilidade, compreensão e apoio que foram fundamentais para a execução e conclusão deste trabalho. Agradeço aos Professores do Centro de Informática da UFPE, por todo o conhecimento compartilhando, que foram fundamentais para minha formação como Mestre. E aos professores do curso de Bacharelado em Ciência da Computação na UFRPE em especial Valmir Macário, que me guiaram para o caminho da pesquisa. E por fim, agradeço aos meus amigos pesquisadores do Centro de Informática da UFPE e da empresa Document Solutions, que estiveram presente em momentos de discussões e descontração que contribuíram na minha jornada no Mestrado.

## RESUMO

O agrupamento de imagens é uma tarefa importante e desafiadora na aprendizagem de máquina. Como na maioria das áreas de processamento de imagens, as últimas melhorias foram obtidas a partir de modelos baseados em aprendizagem profunda. No entanto, os métodos clássicos de aprendizagem profunda têm dificuldade para lidar com transformações espaciais nas imagens de entrada como, por exemplo, escala e rotação. Nesta dissertação, propomos o uso de técnicas de atenção visual para reduzir este problema em métodos de agrupamento profundo de imagens. Nossa hipótese de pesquisa sugere que adicionar camadas de atenção visual em arquiteturas de *deep image clustering* pode tornar os modelos robustos a pequenas variações espaciais nos dados de entrada da rede e melhorar seu desempenho. Avaliamos a combinação de um modelo de agrupamento profundo chamado *Deep Adaptive Clustering* (DAC) com o módulo de atenção visual *Spatial Transformer Networks* (STN). O modelo proposto é avaliado nos conjuntos de dados MNIST e FashionMNIST e superou o modelo de referência nos experimentos realizados. Adicionalmente realizamos diversos experimentos qualitativos para investigar o funcionamento da arquitetura proposta.



**Palavras-chaves**: Agrupamento de Imagens. Redes Neurais Profundas. Atenção Visual. Spatial Transformer Networks. Agrupamento adaptativo

## ABSTRACT

Image clustering is an important but challenging task in machine learning. As in most image processing areas, the latest improvements came from models based on the deep learning approach. However, classical deep learning methods have problems to deal with spatial image transformations like scale and rotation in input images. In this dissertation, we propose the use of visual attention techniques to reduce this problem in deep image clustering methods. Our research hypothesis suggests that adding visual attention layers to deep image clustering architectures can make models more robusts to small spatial variations in the input data of the network and improve their performance. We evaluate the combination of a deep image clustering model called Deep Adaptive Clustering (DAC) with the visual attention module Spatial Transformer Networks (STN). The proposed model is evaluated in the datasets MNIST and FashionMNIST and outperformed the baseline model in the performed experiments. In addition, we performed several qualitative experiments to investigate the operation of the proposed architecture.



**Key-words**: Image Clustering. Deep Neural Networks. Visual Attention. Spatial Transformer Networks. Adaptive Clustering

# LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

## LISTA DE ALGORITMOS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| **AE** | *Autoencoder* |
| **CatGAN** | *Categorical Generative Adversarial Networks* |
| **CDNN** | *ClusteringDNN* |
| **CNNs** | *Convolutional Neural Networks* |
| **ConvDEC-DA** | *Convolutional Deep Embbeded Clustering - Data Augmentation* |
| **DAC** | *Deep Adaptive Clustering* |
| **DCN** | *Deep Clustering Network* |
| **DEC** | *Deep Embbeded Clustering* |
| **DEC-DA** | *Deep Embbeded Clustering - Data Augmentation* |
| **DNN** | *Deep Neural Network* |
| **DNNs** | *Deep Neural Networks* |
| **DRAM** | *Deep Recurrent Atention Model* |
| **EDRAM** | *Enriched Deep Recurrent Atention Model* |
| **GAN** | *Generative Adversarial Networks* |
| **JULE** | *Joint Unsupervised Learning* |
| **MLP** | *Multi Layer Perceptron* |
| **ORB** | *Oriented FAST and Rotated BRIEF* |
| **PCA** | *Principal Component Analysis* |
| **RAM** | *Recurrent Atention Model* |
| **ReLU** | *Rectified Linear Unit* |
| **ResNet** | *Residual Network* |
| **RNN** | *Recurrent Neural Network* |
| **SAE** | *Stacked Autoencoder* |
| **SIFT** | *Scale-Invariant Feature Transform* |
| **ST-DAC** | *Spatial Transformer Deep Adaptive Clustering* |
| **STN** | *Spatial Transformer Network* |
| **SURF** | *Speeded Up Robust Features* |
| **VADE** | *Variational Deep Embedding* |

**VAE** *Variational Autoencoder*

## SUMÁRIO

# 1 INTRODUÇÃO

Os dados utilizados nas atividades de análise estatística ou nas tarefas de aprendizado de máquina normalmente possuem informações *a priori*, como os rótulos de classificação dos dados. Desta forma, torna-se possível o treinamento de um modelo que aprenda o mapeamento entre os dados de entrada e os rótulos das classes. Este processo torna o modelo suficientemente capaz de classificar novos exemplos. Entretanto, na resolução de diversos outros tipos de problemas não dispomos dessa informação, nem conhecemos o padrão de distribuição dos dados gerados por cada classe, sendo esta uma tarefa que representa maior desafio em comparação a anterior. A este tipo de abordagem se dá o nome de aprendizagem não-supervisionada (MARQUES, 2005).

Os métodos de agrupamento ou *clustering* se propõem a tratar problemas dessa natureza. Esses métodos suprem a falta de conhecimento sobre o conjunto de dados, procurando as informações necessárias a partir da similaridade entre os dados, ou na divergência, existente entre os elementos do próprio conjunto.

A principal tática empregada por esses métodos consiste em identificar, no conjunto de dados, grupos de alta densidade de elementos, com padrões em comum, e definir partições no espaço de características que possibilitem a separação de grupos divergentes (MARQUES, 2005). Podemos chamar estes grupos de *clusters*, que pela definição encontrada em (BISHOP, 2006), consistem em "um grupo de pontos de dados, cujas distâncias entre si são pequenas se comparadas com as distâncias para pontos fora do *cluster*".

A atividade de *clustering* é um método de aprendizado não-supervisionado muito utilizado em diversas aplicações de *data mining* e também uma técnica comum para análise de dados estatísticos usada em muitos campos (ZHANG et al., 2007)(SARAVANAN, 2016) (CHENG; ZHANG; CHEN, 2009) (NAGAVI et al., 2013).

Por exemplo, sua utilização é bem frequente nos sistemas de recuperação de imagens baseado em conteúdo. Nestes sistemas uma imagem é passada como entrada, baseada nas características encontradas desta imagem, a aplicação deve retornar todos os exemplos ou um conjunto de imagens, que sejam semelhantes a esta entrada.

Em uma abordagem modesta, o cálculo de similaridade entre a imagem de entrada e todas as imagens do banco de dados seria realizado para que o sistema retorne o conjunto de imagens similar. Assim, o método se torna bastante ineficiente quando a quantidade de imagens no banco de dados é grande. Com a utilização de um método de *clustering* sobre a base de dados, previamente usada como exemplo, o conjunto de imagens é dividido em *clusters* de acordo com as similaridades ou dissimilaridades destes dados. Dessa forma, torna-se fácil comparar as características da entrada com alguns exemplos de cada *cluster* previamente construído, sendo possível retornar o conjunto de imagens com menos passos de processamento.

Em alguns casos, o *clustering* é importante inclusive para o aprendizado supervisionado. Em muitas aplicações reais de classificação de imagens em larga escala, os dados rotulados não estão disponíveis ou não são suficientes para treinar modelos supervisionados, uma vez que o tedioso processo manual de rotulagem de imagens requer muito tempo e trabalho. Uma estratégia amplamente utilizada é aplicar clusterização nos dados não rotulados e, em seguida, usar o mínimo de esforço humano para rotular os grupos e posteriormente seus elementos com base no rótulo do grupo (QI et al., 2007) (SCLAROFF et al., 1999).

O *clustering* de imagens é uma tarefa importante, mas desafiadora, devido à variabilidade entre classes de imagens. Por um longo tempo, técnicas clássicas como *K-means* foram a melhor opção para o *clustering* de imagens (WANG et al., 2012) (WANG et al., 2015). Entretanto, imagens são estruturas de dados com alta dimensionalidade e difíceis de tratar computacionalmente por métodos simples. Logo, se faz necessária a extração de *features* representativas que ressaltem as principais características das imagens processadas, sendo esta uma importante tarefa para tornar o processo de *clustering* mais eficiente. Para superar este problema, foram utilizados métodos como *Principal Component Analysis* (PCA), para realizar a redução de dimensionalidade das representações da imagem e extrair as *features* mais refinadas para realizar a tarefa de *clustering*.

No entanto, nos últimos anos, as *Deep Neural Networks* provaram ser muito eficazes em várias áreas de processamento de imagens e visão computacional, devido a sua grande eficiência em realizar o mapeamento-não linear do conjunto de dados e a função objetivo, eliminando a necessidade de extração de *features* manuais anteriormente desenvolvidas.

Logo, em *clustering* de imagens, inicialmente modelos de *deep learning* foram utilizados como uma etapa de pré-processamento na extração de *features*. O pré-processamento era utilizado nos algoritmos de *clustering* tradicionais. Além disso, estas etapas ocorriam de forma sequencial e separada.

Entretanto, em novos trabalhos a otimização da geração de representações das imagens e a tarefa de *clustering* são realizadas em conjunto, possibilitando a geração de *features* que não apenas destacam as principais características da imagem, mas as tornam adequadas para a tarefa de *clustering*. Além disso, todo o processo de *clustering* automático são realizados pala rede.

Esta abordagem se mostrou mais promissora que as anteriores, melhorando os resultados de *clustering* de imagens até então encontrados na literatura. Destacamos alguns exemplos deste modelos como: *Categorical Generative Adversarial Networks* (CatGAN) (SPRINGENBERG, 2015), *Joint Unsupervised Learning* (JULE) (YANG; PARIKH; BATRA, 2016), *Deep Clustering Network* (DCN) (YANG et al., 2016),*Deep Embbeded Clustering* (DEC) (XIE; GIRSHICK; FARHADI, 2016) e *Deep Adaptive Clustering* (DAC) (CHANG et al., 2017). Este tipo de modelo é referenciado na literatura como *deep image clustering*. Os algoritmos de *deep image clustering* têm atingido o estado-da-arte em vários *benchmarks*

de imagens (MIN et al., 2018).

Nesta categoria, o DAC se destaca dos demais nos resultados obtidos em diversos *benchmarks* conhecidos (MIN et al., 2018)(CHANG et al., 2017). O modelo trata o problema de *clustering* como uma tarefa de classificação binária entre pares, gerando os rótulos a partir das representações da rede e a aplicação de um algoritmo adaptativo. Ao juntar de forma inovadora a tarefa de *clustering* e classificação, esta rede abre a possibilidade para a avaliação de abordagens propostas para tarefas supervisionadas no agrupamento de imagens.

Como comentado anteriormente, as *Deep Neural Networks* (DNNs) são extremamente poderosas. No entanto, elas possuem alguns problemas ao tratar imagens que sofreram transformações espaciais, como a escala e a rotação. A maioria das *Convolutional Neural Networks* (CNNs) utilizam tipicamente camadas de *max-pooling* usando pequenas regiões de agrupamento de pixels (por exemplo, 2 x 2 ou 3 x 3 pixels). A transformação de *max-pooling* permite robustez a variância espacial de apenas uma pequena região da imagem, além disso, os mapas de características intermediários na CNN não são invariantes para diferenças de transformações entre as classes dos dados de entrada.

Diversas abordagens foram propostas, anteriormente aos métodos de *deep learning*, para tornar possível a utilização de recursos invariantes a transformações espaciais e melhorar os resultados na tarefa de *clustering* de imagens. Os métodos mais clássicos usam recursos como *Scale-Invariant Feature Transform* (SIFT)(LOWE, 2004), *Speeded Up Robust Features* (SURF) (BAY et al., 2008) e *Oriented FAST and Rotated BRIEF* (ORB) (RUBLEE et al., 2011) que são invariantes para dimensionamento uniforme, orientação, alterações de iluminação e parcialmente invariantes para distorção como podemos ver em (ZHANG et al., 2017) (ANTONOPOULOS; NIKOLAIDIS; PITAS, 2007) (LEE; AHN; RHEE, 2012) (MIRONICă; IONESCU; VERTAN, 2012) (ZHANG; MIAO, 2014). Trabalhos como estes demostram a importância em obter propriedades de invariância a transformações, na extração de *features* das imagens, assim como nos modelos de *clustering*.

Em pesquisas mais recentes, técnicas mais elaboradas têm sido propostas para tentar tratar esse problema nos modelos de *deep learning*, como o mecanismo de atenção visual *Spatial Transformer Network* (STN) (JADERBERG et al., 2015). Esses módulos podem ser inseridos na arquitetura de rede como uma camada e fornecer ao modelo a capacidade de aprender invariância a escala, rotação e deformações de imagens dadas como entrada (MIN et al., 2018). As STN aprendem as transformações existentes no conjunto de dados e aplicam correções para uma transformação comum de forma a maximizar os resultados da rede na qual está inserida. Além disso, com a utilização das STNs é possível fazer com que a rede localize e foque apenas nas características mais importantes das imagens para a conclusão da tarefa proposta. Esse tipo de camada vem melhorando o desempenho de diversas redes convolucionais em vários *benchmarks* de tarefas supervisionadas. Pressupomos que também podem trazer benefícios aos problemas de *deep image clustering*,

principalmente ao modelo DAC por tratar a tarefa de *clustering* como um problema de classificação.

## 1.1 OBJETIVOS DA DISSERTAÇÃO

Neste trabalho, propomos a investigar o uso de técnicas de atenção visual em modelos de *deep clustering*. Nossa hipótese de pesquisa sugere que adicionar camadas de atenção visual em arquiteturas de *deep image clustering* pode tornar os modelos robustos a pequenas variações espaciais nos dados de entrada da rede e melhorar seu desempenho. Para isso utilizamos como base uma combinação das soluções *Deep Adaptive Clustering* (DAC) (CHANG et al., 2017) e *Spatial Transformer Network* (STN) (JADERBERG et al., 2015).

Como objetivos específicos do trabalho, podemos citar:

- Investigar os principais métodos de *deep image clustering*;
- Investigar os principais métodos de atenção visual no processamento de imagens;
- Avaliar as formas de adicionar camadas STNs em modelos *deep image clustering* como DAC;
- Avaliar experimentalmente o desempenho do modelo proposto em bases de dados públicas comparado ao desempenho da literatura.

Avaliamos nossa abordagem a partir de experimentos com a base de dados MNIST (LECUN et al., 1998a) e Fashion MNIST (XIAO; RASUL; VOLLGRAF, 2017) e comparamos os seus resultados com o modelo base e outros algoritmos bem sucedidos na área de *clustering* de imagens.

## 1.2 CONTRIBUIÇÕES

Ao final deste trabalho, conseguimos contribuir com a literatura com um artigo sobre os temas abordados por esta dissertação chamado *"Improving Deep Image Clustering With SpatialTransformer Layers"* submetido para avaliação na *International Joint Conference on Neural Network 2019*.

Além disso, disponibilizamos todos os códigos com os quais conduzimos nosso experimento*. [1]

## 1.3 ESTRUTURA DA DISSERTAÇÃO

Este trabalho está dividido em 5 capítulos, incluindo este capítulo introdutório.

[1] https://github.com/tvmsouza/ST-DAC

- Capitulo 2: introduz conceitos básicos para compreender a proposta do trabalho. Iniciamos com uma explanação do conceito de *clustering* de dados e suas técnicas, mais especificamente aplicadas a imagens e utilizando técnicas de aprendizado profundo. Em seguida, são apresentados conceitos sobre atenção visual e como são aplicados em visão computacional. Nas duas seções, falaremos sobre os trabalhos relacionados a nossa proposta.

- Capítulo 3: apresenta o modelo proposto, resultante da combinação entre o DAC e STN, critérios e justificativas para as composições apresentadas.

- Capitulo 4: neste capítulo detalhamos as configurações experimentais e as motivações que as conduziram, assim como são discutidos os resultados obtidos por meio das avaliações.

- Capitulo 5: apresenta as considerações finais sobre os resultados obtidos e propõe trabalhos futuros.

# 2 REFERÊNCIAL TEÓRICO

## 2.1 *CLUSTERING* DE DADOS

O *clustering* de dados é uma importante tarefa no aprendizado de máquina, mais especificamente em mineração de dados e que tem por objetivo particionar um conjunto de dados em *clusters* baseando-se na similaridade entre os elementos do grupo.

A similaridade entre os elementos deve preservar duas propriedades: a) maximizar a homogeneidade entre os dados pertencentes a um mesmo grupo; e b) maximizar a heterogeneidade entre elementos de grupos diversos.

Para calcular a similaridade entre dois exemplos de uma base de dados são utilizados cálculos de medidas de similaridade ou dissimilaridade numérica, como a distância euclidiana, cosseno, Mahalanobis. Estas medidas são realizadas sobre os dados originais ou sobre as representações de características extraídas dos exemplos do conjunto de dados (XU; TIAN, 2015).

Existem na literatura diversos métodos de *clustering*, como o baseado em densidade DBSCAN(ESTER et al., 1996), que segue a premissa de que os dados que estão em uma região espacial com alta densidade pertencem ao mesmo *cluster*. Outros se baseiam na ideia de hierarquia, como o *Chameleon* (KARYPIS; HAN; KUMAR, 1999), no qual o principal objetivo é construir uma relação hierárquica entre os dados com o objetivo de formar os *clusters* (XU; TIAN, 2015).

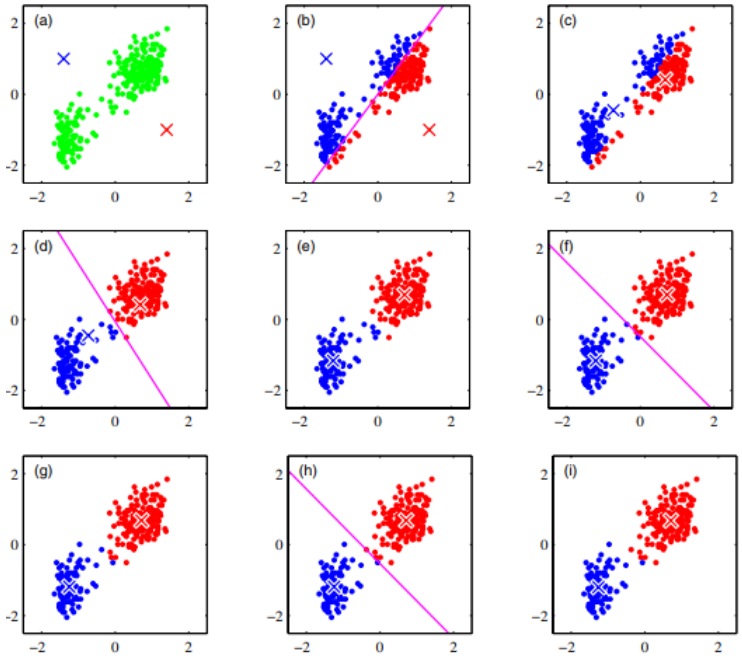
Pertencente a categoria de abordagens baseadas em partições, uma das técnicas mais utilizadas em *clustering* de dados, desde a sua introdução por (LLOYD, 1982) é o *K-means*, que considera os centros dos grupos de dados como os centros dos *clusters* correspondentes. Este método tem sido extensivamente usado sozinho ou juntamente com técnicas de pré-processamento de dados devido à sua simplicidade e eficácia (YANG et al., 2016).

O *K-means* é uma técnica adequada para realizar o *clustering* de dados em um cenário em que as amostras são uniformemente espalhadas em torno de alguns elementos ou centroides, ou seja, pontos que representam o objeto central de cada grupo. Seu funcionamento necessita que o número de *clusters* $K$ seja passado por parâmetro pelo projetista do modelo. Dada esta informação, $K$ centroides iniciais são definidos aleatoriamente. Em seguida, cada exemplo da base de dados é classificado de acordo com a distância calculada entre este exemplo e os centroides. O exemplo é atribuído ao *cluster* no qual a distância até o seu centro seja menor. Os centroides de cada grupo são então atualizados a partir do cálculo da média entre as amostras. Excluindo a etapa de inicialização aleatória dos centroides, todo o processo seguinte é repetido até atingir um número máximo de iterações ou atingir um parâmetro de convergência.

A Figura 1 ilustra as etapas de execução do algoritmo *K-means*. Nela, vemos a apli-

cação do algoritmo sobre um conjunto de dados redimensionado para ser exibido em um espaço euclidiano bidimensional. O objetivo é a formação de dois *clusters* a partir dos dados de entrada. No segmento (a) os dados estão dispostos no espaço como pontos verdes, os dois centroides inicializados aleatoriamente, são representados por um $X$ vermelho e azul respectivamente. Em (b) é atribuído a cada ponto um *cluster*, de acordo com a proximidade aos centroides. No quadro (c) os novos centroides são recalculados a partir da média dos valores dos pontos de cada *cluster*. Este processo segue da mesma forma nas demais iterações como podemos ver nos segmentos (d) a (h) até atingir a convergência em (i) e os *clusters* serem formados.

Figura 1 – Etapas de execução algoritmo *K-means* sobre um conjunto de pontos.




Fonte: (BISHOP, 2006)

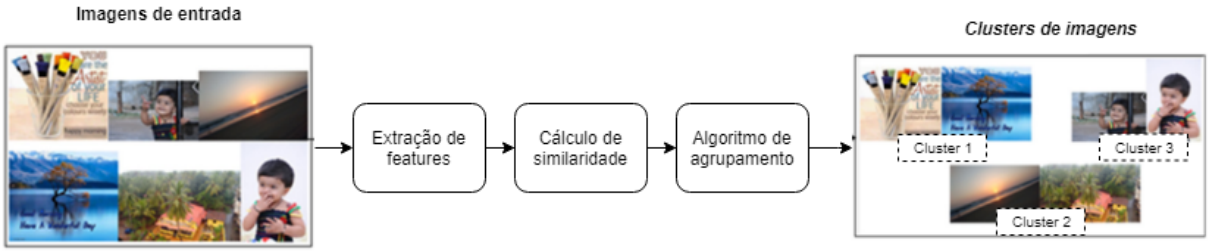
Neste exemplo, ilustramos a aplicação de uma técnica de *clustering*, sobre um conjunto de pontos em um plano, entretanto, técnicas de *clustering* como as citadas anteriormente podem ser utilizadas para tratar dados de qualquer dimensionalidade ou formato, sejam eles no formato numérico ou texto, ou até mesmo objetos de multimídia, como áudio, imagens e vídeos. Em estruturas de dados multimídia, geralmente estes métodos são utilizados em aplicações que realizam a segmentação do conteúdo de uma única instância ou arquivo, como os pixeis de uma determinada região da imagem, ou são utilizados para realizar o *clustering* de várias instâncias, como nas aplicações de recuperação de imagem ou indexação automática de dados.

Neste trabalho, vamos focar na tarefa de *clustering* de múltiplas imagens em subconjuntos, logo, os detalhes fornecidos no texto são inerentes a este campo de estudo.

### 2.1.1 *Clustering* clássico de imagens

As técnicas de *clustering* de imagens normalmente utilizam a mesma estrutura das técnicas tradicionais de *clustering*, se baseando em um alto nível de abstração, como utilizados em outros tipos de dados, como apresentado na Figura 2. As etapas são realizadas de forma sequencial e independente. Inicialmente, as entradas são obtidas a partir do pré-processamento das *features* extraídas da imagem. Em seguida é realizado o cálculo da medida de similaridade entre as representações, e por fim é empregado o algoritmo de *clustering* tradicional, que é incumbido de encontrar e refinar os possíveis *clusters* dos dados. A entrada dada ao sistema é um conjunto de imagens e a saída é o conjunto de rótulos, que atribui, para cada imagem de entrada o *cluster* ao qual pertence.

Figura 2 – Etapas de um sistema de *clustering* de imagens.




Fonte: Adaptada de (WAZARKAR; KESHAVAMURTHY, 2018)

Comparado a um sistema de *clustering* de imagens, a aplicação das técnicas de *clustering* tradicionais em conjuntos de dados mais básicos, compostos apenas por números ou texto, possui uma demanda muito menor de poder computacional, logo podem ser utilizadas representações simples dos dados para o *clustering*.

Um dado do tipo imagem, porém, é uma estrutura repleta de informações. Uma única imagem é composta por uma matriz numérica, na qual cada valor representa a intensidade de um pixel a ser exibido na tela. Estas estruturas podem ter ainda diversos tamanhos de resolução. Sendo assim, uma simples imagem pode conter milhares de pixels em sua composição, nos quais, independente da similaridade de conteúdo, somente a relação espacial entre eles pode definir sua correta representação. Estruturas de dados de alta dimensionalidade como essa são muito mais difíceis de tratar computacionalmente por métodos de aprendizado de máquina, como *clustering*, quando comparado a tipos mais básicos de dados.

Para realizar o *clustering* com esse tipo de dado é necessário extrair uma descrição de forma mais detalhada possível das imagens que se deseja agrupar. Essas descrições podem ser chamadas de *features*, as quais são representações que descrevem a imagem a partir de características descorrelacionadas e deseja-se que tenham propriedades invariantes a transformações como escala, translação e rotação(MARQUES, 2008).

Logo, no processo de *clustering* de imagens uma etapa importantíssima é a extração de *features*. E seu principal objetivo é transformar os dados com baixo nível de informação

das imagens e de alta dimensionalidade em uma quantidade gerenciável de propriedades relevantes, reduzindo a complexidade do descritor de *feature* e o tornando mais robusto (SILESHI; GAMBACK, 2009).

Muito esforço tem sido empregado na implementação de métodos de extração de *features*. Durante anos, diversas técnicas clássicas de pré-processamento de imagem foram empregadas para construir uma representação que pudesse ser utilizada nos algoritmos de aprendizado de máquina.

Um exemplo de *features* bem utilizadas são os descritores de forma que extraem informações de dimensão e área dos objetos na imagem. Uma diversidade de outros métodos extraem características importantes como a textura e a partir delas conseguem distinguir áreas que apresentem as mesmas características de padrões e cores de uma determinada região.

Também é possível a extração de descritores locais como SIFT (LOWE, 2004) ou SURF (BAY et al., 2008), estes ainda bem utilizados, são invariantes a transformações de redimensionamento, rotação ou iluminação. Outros métodos foram idealizados para ressaltar características importantes das imagens e reduzir a dimensionalidade da representação das mesmas como a *Principal Component Analysis* (PCA).

Existe uma vasta gama de estudos e técnicas desta área. Todos esses descritores podem ser utilizados sozinhos ou até mesmo uma combinação entre todas essas representações pode ser feita para construir *features* bem representativas dos dados de entrada.

Motivadas pelo sucesso das *Deep Neural Networks* (DNNs) na aprendizagem supervisionada e não supervisionada de dados, foram propostas nos últimos anos técnicas mais modernas que utilizam *deep learning* na extração de *features*. Estas abordagens também são amplamente utilizadas para redução de dimensionalidade das imagens de entrada. Modelos como, o *Autoencoder* (AE) (BENGIO et al., 2006) são utilizados para este fim. Um *Autoencoder* é uma rede neural totalmente conectada e pode ser composta por uma arquitetura *deep*. O objetivo da rede é aprender a função de identidade, ou seja, reproduzir os dados de entrada na saída. Além disso, a rede contém uma única camada oculta que possui um número de nós significativamente menor que a entrada. Sendo assim, durante a fase de treinamento, o *Autoencoder* tenta encontrar uma representação compactada dos dados de entrada. Estas representações podem ser usadas em outras aplicações de *machine learning* como *features*. Outra variação destes modelos são os *Stacked Autoencoder* (SAE) (VINCENT et al., 2010) que conectam a saída da camada oculta compactada à entrada de um outro *Autoencoder* para formar arquiteturas profundas. Esses modelos se mostram úteis na realização de redução de dimensionalidade de imagens.

Entretanto, os métodos de extração de *features* citados anteriormente muitas vezes são propostos para serem utilizados em diversos tipos de tarefas de visão computacional. Eles ressaltam alguma característica específica da imagem, reduzem a dimensionalidade dos dados e oferecem informações globais e locais sobre a imagem, que são boas propriedades

a se esperar de uma *feature* a ser utilizada em técnicas de *machine learning* em geral. Entretanto, estas *features* geradas não são direcionadas para a utilização em tarefas de *clustering* de forma a otimizar os resultados do algoritmo de agrupamento.

Além disso, como vimos, as abordagens clássicas de *clustering* de imagens realizam a extração de *features* das imagens e o método de *clustering* separadamente. Mesmo algumas técnicas de extração de *features* que utilizam DNNs e que aprendem uma compactação dos dados de entrada eficiente, tratam suas redes neurais profundas como um estágio de pré-processamento das imagens, o qual é separado do método de *clustering* utilizado.

Desta forma, uma vez que o objetivo de agrupar os dados não é incorporado ao processo de aprendizagem das redes neurais profundas ou na extração das *features*, as representações aprendidas por estes métodos não necessariamente são direcionadas à tarefa de *clustering* (YANG et al., 2016).

### 2.1.2 *Deep Image Clustering*

Em novos trabalhos de clustering de imagens é proposto que a otimização do método de *clustering* seja realizada junto com o método de redução de dimensionalidade e extração de *features*. A abordagem consegue gerar assim representações latentes dos dados mais adequadas a realização da tarefa de *clustering* do que as obtidas pela estratégia clássica de *clustering* de imagens.

Sendo aasim, técnicas como *Deep Neural Network* passaram a tratar a tarefa de gerar representações latentes adequadas para *clustering* e realizar a própria tarefa de *clustering* em si, retornando os *clusters* encontrados no conjunto de dados. Entretanto, para realizar esta tarefa, é necessário que propriedades de *clustering* sejam adicionadas às funções de custo e à arquitetura da rede. Logo, redes de *deep image clustering* possuem uma função de custo de *clustering* $L_c$ que contém os centroides dos *clusters*, e são responsáveis pela atribuição do *cluster* às imagens de entrada, além de trazer propriedades de grupamento no processo de geração de *features*. Desta forma, após a etapa de treinamento, os *clusters* podem ser obtidos diretamente da rede.

Estes modelos, em geral, também possuem a função de custo auxiliar $L_n$ que forçam as redes a criar representações mais verossímeis e não triviais dos dados, as quais ajudam no processo de *clustering*, mas não realizam a clusterização individualmente. Essas funções geralmente são encontradas em modelos de *deep image clustering* compostos por um *Autoencoder*, *Generative Adversarial Networks* (GAN) ou *Variational Autoencoder* (VAE), nestes modelos as funções de reconstrução da imagem de entrada são usadas como a função de custo auxiliar.

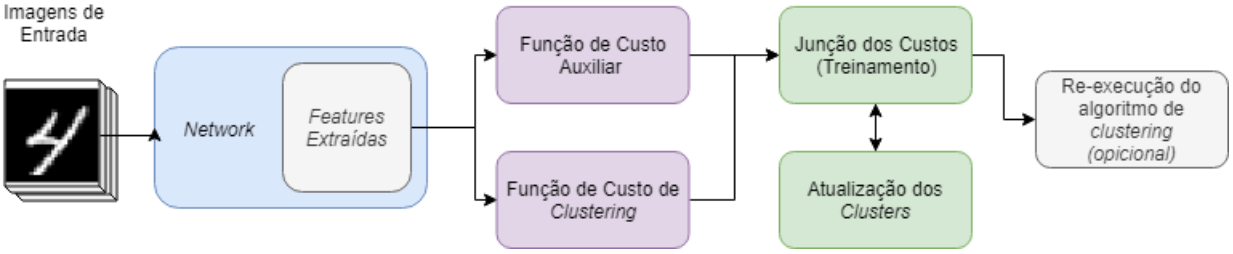
Em linhas gerais, estas duas funções $L_n$ e $L_c$ são combinadas em *deep image clustering* como mostrado na Equação 2.1, na qual o parâmetro $\lambda \in [0, 1]$ trata de realizar o balanceamento entre as duas funções. Assim, elas são capazes de realizar a tarefa de *clustering*

ao mesmo tempo em que, no processo de treinamento, são capazes de gerar *features* muito mais adequadas e direcionadas à tarefa de *clustering* (MIN et al., 2018).

$$L = \lambda L_n + (1 - \lambda) L_c \tag{2.1}$$

As abordagens propostas na literatura utilizam as mais diversas arquiteturas, variando na estrutura e tipos das redes profundas, métodos de *clustering* e funções de custo. Podemos exemplificar o funcionamento e a composição de um *framework* de *deep image clustering* a partir da Figura 3.

Figura 3 – Segmentos de um sistema de *deep image clustering*.




Fonte: Adaptada de (ALJALBOUT et al., 2018)

Na Figura 3 podemos analisar os segmentos que constituem o sistema da maioria dos modelos de *deep image clustering*, de acordo com (ALJALBOUT et al., 2018). O sistema é divido em segmentos bem definidos, que podem ser compostos por diversas técnicas.

Observamos primeiramente o bloco da rede, na qual as imagens são passadas para a camada de entrada da rede neural profunda. Esta se encarrega de gerar as *features* latentes do objeto de entrada e retorna pela camada de saída as *features* extraídas da imagem em dimensões reduzidas. Neste segmento, diversos modelos de rede podem ser utilizados, como, por exemplo, a rede *Multi Layer Perceptron* (MLP), *Convolutional Neural Networks*, *Autoencoder*, *Stacked Autoencoder*. Em algumas abordagens de modelos *deep image clustering* as *features* extraídas podem ser combinadas com as saídas das demais camadas intermediárias da rede. Com a utilização desta técnica, objetiva-se gerar uma representação mais robusta do objeto de entrada.

Em seguida, as *features* são passadas para os blocos de otimização, no qual os custos da iteração são calculados. O primeiro bloco corresponde ao ramo da função de custo de *clustering* $L_c$, responsável por gerar as representações adequadas ao agrupamento. O segundo é constituído pelo ramo com a função de custo auxiliar $L_n$. Este último é responsável por realizar algumas restrições no processo de otimização da rede, assim como evitar o retorno de respostas indevidas.

Diversas funções de custo de *clustering* são encontradas nos modelos de *deep image clustering*. Como exemplo, podemos citar a *K-means loss*, utilizada em (YANG et al., 2016),

que segue descrita na Equação 2.2.

$$L(\theta) = \sum_{i=1}^{N} \sum_{k=1}^{K} s_{ik} \left\| z_i - \mu_k \right\|^2 \tag{2.2}$$

Nesta equação, $z_i$ representa a *feature* extraída, $\mu_k$ são os centroides e $s_{ik}$ é uma variável booleana que atribui $z_i$ a $\mu_k$. Esta função garante que as *features* extraídas são adequadas para a execução do *K-means* posteriormente e que os pontos de dados estão distribuídos uniformemente em torno dos centroides. Logo, o objetivo dessa equação é minimizar a distância entre as *features* extraídas e o centroide, melhorando os resultados posteriores de uma aplicação do algoritmo *K-means*.

As funções de custo auxiliar geralmente são empregadas, por modelos de *deep image clustering* compostos por uma DNN extratora como VAE, GAN ou *Autoencoder* que possuem uma função de reconstrução da imagem de entrada como a exibida na Equação 2.3.

$$L = d_{\mathrm{AE}} \left( x_i, f\left( x_i \right) \right) = \sum \left\| x_i - f\left( x_i \right) \right\|^2 \tag{2.3}$$

Nesta função, $x_i$ é a imagem de entrada é $f(x_i)$ a imagem reconstruída pelo modelo, sendo $d_{\mathrm{AE}}$ nada mais que a distância entre a imagem de entrada e sua reconstrução.

Após esta etapa, os custos obtidos previamente são combinados no segmento de junção e podem ser compostos por diversos métodos igualmente. Geralmente, são combinados como mostrado na Equação 2.1. Neste estágio, o custo total da iteração é calculado e a rede é treinada dependendo do método utilizado.

A atualização dos *clusters* e centroides pode acontecer durante o processo de treinamento da rede e as atribuições dos *clusters* retornam como uma probabilidade gerada pela mesma. Em outras abordagens, a atualização dos *clusters* pode ser feita em um módulo separado do algoritmo, independente da rede, e após um determinado número de iterações.

Em último estágio, algumas estratégias propõem executar novamente o algoritmo de *clustering* já treinado, para aprimorar os resultados, outras aproveitam os conceitos de *transfer learning* testando o modelo em outra base de dados.

Os modelos de *deep image clustering* da literatura, relacionados a este trabalho de dissertação, são constituídos por diversos tipos de configurações. Encontramos nos métodos várias abordagens que modificam cada um destes blocos e estratégias citadas anteriormente. Diversas outras funções de custo e outras alterações destes segmentos podem ser encontradas em (ALJALBOUT et al., 2018). Inclusive, a quantidade de arquiteturas de redes profundas utilizadas em *deep image clustering* são bem variadas e distintas, como podemos ver em (MIN et al., 2018).

Muitos trabalhos em *deep image clustering* obtiveram resultados marcantes ou se tornaram importantes pela forma de tratar o problema de agrupamento. Todos estes métodos se relacionam diretamente com nossa proposta. Destacamos em *deep image clustering*, o

trabalho *Deep Clustering Network* (DCN) (YANG et al., 2016), que combina uma rede AE pré-treinada com o algoritmo *k-means*. Em conjunto, o modelo otimiza a função de custo auxiliar de reconstrução e de *clustering* a partir da junção dos métodos citados anteriormente nesta seção.

O *Joint Unsupervised Learning* (JULE) (YANG; PARIKH; BATRA, 2016) utiliza um módulo de *clustering* hierárquico e redes convolucionais profundas para gerar as representações das imagens, os dois métodos também são otimizados conjuntamente.

Outros modelos interessantes são baseados em *Generative Adversarial Networks* (GAN) (GOODFELLOW et al., 2014) e *Variational Autoencoder* (VAE) (KINGMA; WELLING, 2013) como *Categorical Generative Adversarial Networks* (CatGAN) (SPRINGENBERG, 2015) e *Variational Deep Embedding* (VADE) (JIANG et al., 2017) que são capazes de gerar novas imagens relacionadas aos grupos aprendidos, além de realizar o *clustering* das imagens.

Um dos métodos mais representativos de aprendizagem profunda e que atrai muita atenção da literatura *deep image clustering* é o *Deep Embbeded Clustering* (DEC) (XIE; GIRSHICK; FARHADI, 2016). O método realiza um pré-treinamento em um *Stacked Autoencoder*, depois organiza as camadas da arquitetura para formar um *Deep-Autoencoder*, no qual se realiza um *fine-tuning*. Em seguida, a parte do *decoder* é removida da rede e a saída do *encoder* serve como *feature* para o módulo de *clustering*. A rede é otimizada usando o método *hardness-loss clustering* para atribuir os rótulos aos exemplos iterativamente. Esse modelo é referência para avaliação de novos modelos e experimentos com *clustering* de imagens (MIN et al., 2018).

Devido a sua importância na área de *deep image clustering*, outros métodos foram propostos com base no DEC, como o *Deep Embbeded Clustering - Data Augmentation* (DEC-DA) (GUO et al., 2018). Neste modelo, os autores primeiro treinam um AE no qual a entrada são as imagens com *data augmentation*. Com a rede treinada e capaz de gerar *features* representativas, são empregados no treinamento a função de custo de *clustering* e a função auxiliar de reconstrução das *features* do AE combinadas. Neste processo, as *features* são geradas pelo *decoder* a partir das imagens com *data augmentation*. Os centroides são calculados a partir das representações geradas pelo mesmo *decoder*, entretanto, neste momento o *decoder* recebe as imagens originais sem transformações. Toda a rede segue otimizada em conjunto. Neste trabalho, são propostas diversas arquiteturas de AE, entretanto a que mais se destaca é a formada por AE convolucionais *Convolutional Deep Embbeded Clustering - Data Augmentation* (ConvDEC-DA). A estratégia de incluir *data augmentation* na reconstrução do AE se mostrou bastante eficiente e atingiu o estado-da-arte em várias bases de dados.

Todos os modelos citados apresentam ótimos resultados e estratégias inovadoras para abordar o problema de *deep image clustering*. Entretanto, nenhuma destas estratégias utilizam STNs em sua composição. Escolhemos o algoritmo DAC como base para nossos experimentos, por seu valor histórico e resultados em diversos *benchmarks*, este modelo é

detalhado na próxima seção.

#### 2.1.2.1 *Deep Adaptive Clustering* - DAC

O DAC é um modelo de *deep image clustering*, baseado em uma rede convolucional de estágio único ou seja, para realizar o *clustering* das imagens não é necessário etapas prévias de pré-treinamento no modelo convolucional, nem estágios adicionais de execuções sequenciais de módulos de *clustering* independentes.

De acordo com as categorias em que se enquadram os métodos de *deep clustering* apresentados por(MIN et al., 2018), DAC se enquadra no grupo das *ClusteringDNN* (CDNN) assim como o algoritmo JULE (YANG; PARIKH; BATRA, 2016) já apresentado anteriormente.

Diferente de grande parte dos modelos de *deep image clustering* propostos, DNNs presentes nesta categoria excluem a necessidade de uma função de custo auxiliar e são apenas otimizadas pela função de custo de *clustering* $L_c$, como na Equação 2.4. Entretanto, estas funções devem ser bem definidas pois são unicamente responsáveis por otimizar a geração de boas *features* das imagens e realizar o processo de *clustering*.

$$L = L_c \tag{2.4}$$

O modelo apresenta uma característica um tanto inovadora ao tratar o problema de *clustering*, abordando o problema de uma maneira diferente dos demais modelos de *deep image clustering*. Geralmente, outros métodos utilizam técnicas mais específicas da área de agrupamento como *hierarquical clustering* ou *K-means*. O DAC propõe tratar a tarefa de *clustering* como uma tarefa de classificação binária entre pares. Desta forma os pares de imagens podem ser classificados no sistema como pertencentes a um mesmo grupo, ou a grupos diferentes, dependendo de suas similaridades.

Em um problema de classificação, são necessários os rótulos das classes para treinar o modelo. Para contornar esta situação, o método propõe obter os rótulos a partir das próprias *features* extraídas pela rede. Para isso, emprega algumas restrições sobre a saída do modelo e assim consegue gerar *features* adequadas para *clustering*.

Como descrito em (CHANG et al., 2017), a classificação binária em pares funciona da seguinte forma:

Denotamos o conjunto de imagens não-rotuladas apresentadas para o agrupamento como $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{n}$ e o parâmetro de inicialização $K$ que representa o número de grupos formados ao término da execução do método, a partir do conjunto de dados de entrada. A variável $x_i$ é a i-ésima imagem do grupo de dados. Logo, num problema de classificação, precisamos ter algum tipo de rótulo para as classes do problema. Denotamos então o conjunto de dados de treinamento como, $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{x}_j, r_{ij})\}_{i=1,j=1}^{n}$ no qual $x_i$ e $x_j$ são diferentes imagens de entrada não-rotuladas presentes no conjunto $X$ e a variável $r_{ij} \in Y$ é uma variável de saída binária desconhecida, a qual indica, se $r_{ij} = 1$, as imagens $x_i$ e

$x_j$ pertencem ao mesmo grupo, caso contrário, se $r_{ij} = 0$, as imagens pertencem a grupos diferentes. A partir das informações que obtemos até agora, podemos entender a função de otimização do DAC como na Equação 2.5.

$$\min_{\mathbf{w}} \mathbf{E}(\mathbf{w}) = \sum_{i,j} L\left(r_{ij}, g\left(\mathbf{x}_i, \mathbf{x}_j; \mathbf{w}\right)\right) \tag{2.5}$$

Nesta equação, $L$ representa a função de custo entre $r_{ij}$ e $g\left(\mathbf{x}_i, \mathbf{x}_j; \mathbf{w}\right)$ a similaridade estimada entre o par de imagens, $w$ representa os parâmetros do modelo que gera as representações de $x_i$ e $x_j$.

A descrição de $L$ é apresentada abaixo na Equação 2.6, os elementos citados compõem a função *binary cross-entropy*.

$$L\ r_{ij}, g\left(\mathbf{x}_i, \mathbf{x}_j; \mathbf{w}\right) = -r_{ij} \log\left(g\left(\mathbf{x}_i, \mathbf{x}_j; \mathbf{w}\right)\right) - \left(1 - r_{ij}\right) \log\left(1 - g\left(\mathbf{x}_i, \mathbf{x}_j; \mathbf{w}\right)\right) \tag{2.6}$$

Neste cenário, (CHANG et al., 2017) afirma que o DAC não garante que os *clusters* são formados apenas a partir da similaridade entre os pares de representações. Para contornar esta situação, aplica-se uma restrição de *clustering* sobre as representações das imagens extraídas pela rede convolucional. Sendo, $\mathcal{L} = \left\{l_i \in R^k\right\}_{i=1}^{n}$ o conjunto das representações extraídas pela rede do conjunto de imagens e $l_i$ a representação da imagem $x_i$, acompanhamos abaixo o formato da restrição na Equação 2.7.

$$\forall i, \left\|\mathbf{l}_i\right\|_2 = 1, \text{ and } l_{ih} \geq 0, h = 1, \cdots, k \tag{2.7}$$

Nesta equação, $\|\cdot\|_2$ representa a norma $L_2$ de um vetor e $l_{ih}$ é o valor na posição $h$ do vetor de representações extraídos pela rede convolucional $f$. Sendo assim, a função cosseno de similaridade pode ser utilizada no modelo. Logo $g\left(\mathbf{x}_i, \mathbf{x}_j; \mathbf{w}\right)$ pode ser descrita como, $g\left(\mathbf{x}_i, \mathbf{x}_j; \mathbf{w}\right) = f\left(\mathbf{x}_i; \mathbf{w}\right) \cdot f\left(\mathbf{x}_j; \mathbf{w}\right) = \mathbf{l}_i \cdot \mathbf{l}_j$, onde "." é o produto interno entre as representações extraídas e $\mathbf{w}$ os parâmetros da rede. A partir destas definições o modelo DAC pode ser reescrito como na Equação 2.8.

$$\begin{aligned} &\min_{\mathbf{w}} \mathbf{E}(\mathbf{w}) = \textstyle\sum_{i,j} L\left(r_{ij}, \mathbf{l}_i \cdot \mathbf{l}_j\right) \\ &\text{s.t. } \forall i, \left\|\mathbf{l}_i\right\|_2 = 1, \text{ and } l_{ih} \geq 0, h = 1, \cdots, k \end{aligned} \tag{2.8}$$

A partir do teorema apresentado e comprovado em (CHANG et al., 2017) o autor afirma que a restrição empregada no modelo possibilita que as imagens sejam agrupadas automaticamente a partir das *features* retornadas pela rede. Isto indica que as *labels features* extraídas da rede são $K$ diversos vetores unitários.

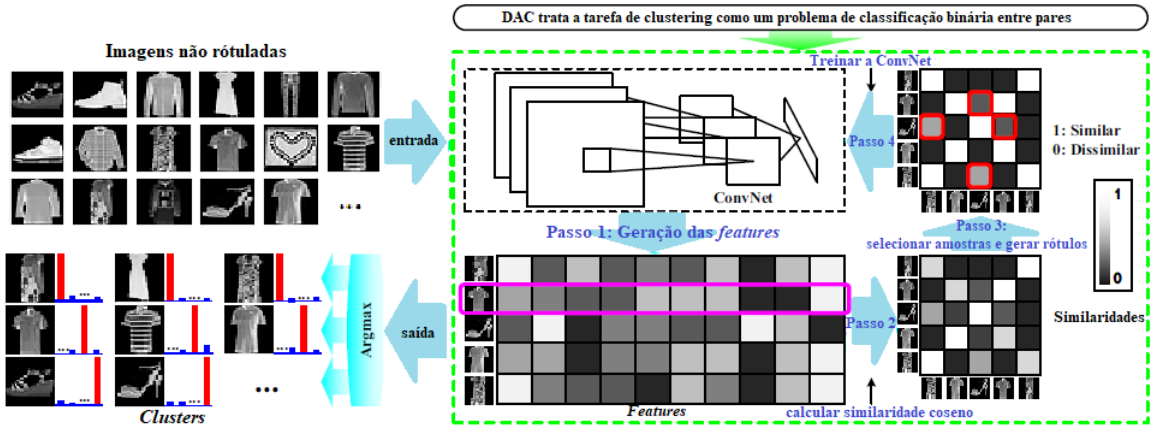
O modelo convolucional utilizado no DAC é a rede *All-ConvNets* (SPRINGENBERG et al., 2014). Esta arquitetura, nos estágios iniciais do processo de treinamento, ainda não é capaz de distinguir muito bem a similaridade entre as imagens com alta *variância* intra-classe. Sendo $Y$ os rótulos de classificação desconhecidos no *clustering* de imagens

e gerados a partir das próprias *features*, é necessário empregar um método adaptativo de seleção de exemplos de treino, escolhendo amostras com um alto nível de confiança nas predições. Este método consiste em selecionar primeiramente para treinamento os rótulos de classificações a partir dos pares que apresentam similaridades ou dissimilaridades mais altas percebíveis durante as primeiras execuções, com maior probabilidade de o resultado da classificação entre o par estar correta. Com o progresso do treinamento a rede se torna mais robusta, logo, amostras de pares de treinamento com maior nível de incerteza são apresentadas gradualmente para que a rede aprenda padrões de *clustering* mais refinados. Assim, baseado na similaridade entre os pares, os rótulos de treinamento são gerados como exemplificado na Equação 2.9.

$$r_{ij} = \begin{cases} 1, \text{ se } l_i \cdot l_j \geq u(\lambda) \\ 0, \text{ se } l_i \cdot l_j < l(\lambda), \quad i, j = 1, \cdots, n, \\ \text{Nada, em outro caso} \end{cases} \tag{2.9}$$

no qual $r_{ij}$ é o rótulo de treinamento gerado pelas dissimilaridade entre as representações, $l_i.l_j$ é o produto interno usado para calcular a distância entre as representações, $\lambda$ é um parâmetro adaptativo que controla a seleção de amostras apresentadas para treino, $u(\lambda)$ é o *threshold* para selecionar amostras de pares similares, $l(\lambda)$ é usado para selecionar as amostras de pares dissimilares e "Nada" significa que a amostra $(x_i, x_j, r_{ij})$ não é apresentada para o processo de treinamento.

Figura 4 – Etapas de geração de *features*, rótulos e seleção de amostras para treinamento do método *Deep Adaptive Clustering* - DAC.




Fonte: Adaptada de (CHANG et al., 2017)

Na Figura 4, podemos acompanhar o processo de seleção de amostras. Inicialmente, o DAC recebe o conjunto de imagens de entrada. Em seguida, as *features* das imagens são geradas a partir da rede convolucional com restrições de *clustering* na saída. A partir do cálculo da distância, é gerada uma matriz de similaridade entre os pares e, no passo seguinte, são selecionados para treinamento apenas os pares com um alto nível de confiança

de similaridade ou dissimilaridade. Os pares não selecionados, destacados pela marca em vermelho na figura, são excluídos do processo de treinamento da rede nesta iteração.

Com o decorrer do processo de treinamento, o parâmetro $\lambda$ vai gradualmente crescendo, ocasionando a diminuição de $U(\lambda)$ e o aumento de $L(\lambda)$, satisfazendo os seguintes critérios, $l(\lambda) <= u(\lambda)$ até os limiares atingirem a igualdade $l(\lambda) = u(\lambda)$ e todos os exemplos de treino serem exibidos à rede. Sendo assim, podemos descrever o modelo DAC, até o momento, através da Equação 2.10:

$$\min_{\mathbf{w},\lambda} \mathbf{E}(\mathbf{w}, \lambda) = \sum_{i,j} v_{ij} L\left(r_{ij}, \mathbf{l}_i \cdot \mathbf{l}_j\right) + u(\lambda) - l(\lambda) \tag{2.10}$$

O parâmetro $v_{ij}$ controla a seleção das imagens que são utilizadas no treinamento. Quando $r_{ij} \in \{0, 1\}$ ,$v_{ij} = 1$ a imagem é levada em consideração no cálculo do custo e otimização da rede, caso contrário $v_{ij} = 0$ o exemplo é desconsiderado. A variável $u(\lambda) - l(\lambda)$ é um termo de penalidade para o número de exemplos. Com o decréscimo do termo de penalidade, mais amostras são adicionadas ao treinamento. A partir destas informações, se pode chegar ao algoritmo *Deep Adaptive Clustering*(1):

**Algoritmo 1:** *Deep Adaptive Clustering*

**Entrada:** Conjunto de imagens $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{n}, f_{\mathbf{w}}, \lambda, u(\lambda), l(\lambda), \eta, m$
**Saida:** Rótulo do *Cluster* $c_i$ de $x_i \in X$

```
Inicializar w aleatoriamente
repetir
for all k ∈ {1, 2, ··· , ⌊n/m⌋} do
    Selecionar batch X_k de X; m imagens por batch
    Selecionar amostras de treino a partir de X_k a partir da Equação( 2.9);
    Calcular o parâmetro indicador v;
    Atualizar w minimizando Equação( 2.13);
end for
Atualizar λ de acordo com a Equação( 2.15;)
até l(λ) > u(λ)
for all x_i ∈ X do
    I_i := f(x_i; w)
    c_i := arg max_h (l_ih)
end for
```

As funções de mapeamento das restrições são utilizadas como uma camada logo após a saída de rede *All-ConvNets* e são descritas nas Equações 2.11 e 2.12.

$$L_h^{out} := \exp^{L_h^{in} - \max_h\left(L_h^{in}\right)}, h = 1, \cdots, k \tag{2.11}$$

$$L_h^{out} := \frac{L_h^{out}}{\left\|\mathbf{L}^{out}\right\|_2}, h = 1, \cdots, k \tag{2.12}$$

Nestas equações, $L^{in}$,$L^{out} \in R$ são a entrada e a saída das camadas de restrição, respectivamente. $L_h^{in}$ e $L_h^{out}$ representam o i-ésimo elemento de $L^{in}$ e $L^{out}$. Desta forma, todos

os elementos de $L^{out}$ são mapeados entre [0,1] pela Equação 2.11 e $L^{out}$ é simultaneamente limitada a um vetor unitário. Sendo assim, as saídas da rede invariavelmente satisfazem a restrição de *clustering*.

A otimização de $w$ e $\lambda$ é realizada alternadamente, uma vez que $r$ e $v$ são obtidos e $\lambda$ é fixado. A função de otimização do DAC definida como na Equação 2.13.

$$\min_{\mathbf{w}} \mathbf{E}(\mathbf{w}) = \sum_{i,j} v_{ij} L\left(r_{ij}, f\left(\mathbf{x}_i; \mathbf{w}\right) \cdot f\left(\mathbf{x}_j; \mathbf{w}\right)\right) \tag{2.13}$$

A partir do momento em que as informações $r$ e $v$ estiverem disponíveis, a Equação 2.13 torna-se um problema de aprendizagem supervisionada e o algoritmo *Backpropagation* pode ser utilizado para atualizar a rede. As amostras de imagens são apresentadas à rede por meio da seleção *batchs* de amostras do conjunto $X$ de imagens. Visto que é computacionalmente custoso calcular e armazenar as matrizes com os rótulos $r$ e a variável de seleção $v$ para todas as imagens em uma única vez. Sendo assim, o modelo DAC pode ser simplificado da seguinte forma quando a rede $w$ é fixada:

$$\min_{\lambda} \mathbf{E}(\lambda) = u(\lambda) - l(\lambda) \tag{2.14}$$

O parâmetro $\lambda$ neste processo é atualizado através do algoritmo de gradiente descendente a cada iteração da rede:

$$\lambda := \lambda - \eta \cdot \frac{\partial \mathbf{E}(\lambda)}{\partial \lambda} \tag{2.15}$$

As *features* são idealmente vetores unitários, logo as imagens podem ser agrupadas a partir da seguinte operação:

$$c_i := \arg\max_{h} \left(l_{ih}\right), h = 1, \cdots, k \tag{2.16}$$

Onde, nesta operação, $c_i$ é o cluster atribuído à imagem $x_i$. Esta etapa é necessária pois na prática as *features* retornadas necessariamente não são vetores unitários. Os autores afirmam que a causa deste problema é a propriedade não convexa das redes convolucionais e a dificuldade em achar um ótimo global durante o treinamento. Sendo assim, para contornar essa situação, é selecionada a maior resposta da *feature* e atribuída ao *cluster*.

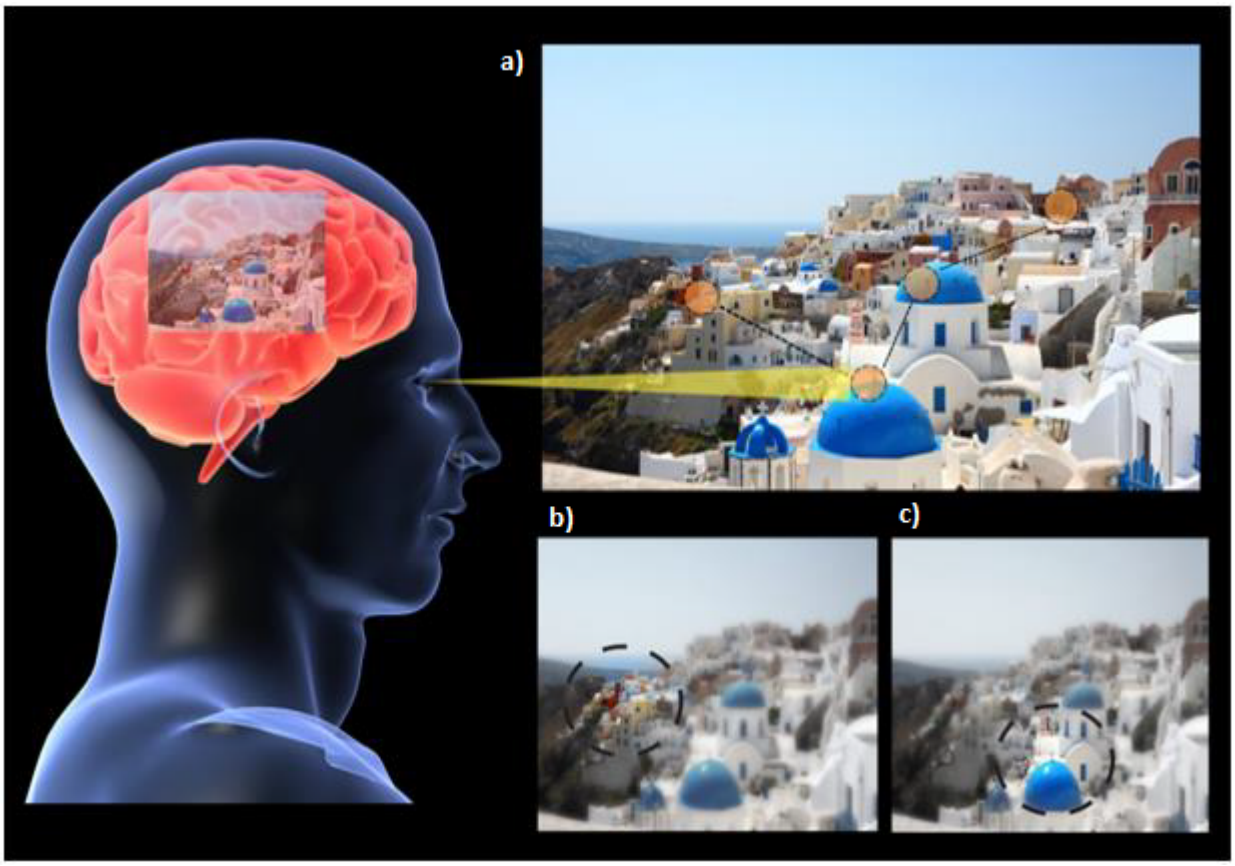
O DAC foi selecionado para nossa proposta por apresentar excelentes resultados na avaliação em diversos *benchmarks* conhecidos. O modelo trata a tarefa de clusterização como um problema de classificação binário, fato que abre a gama de possibilidades na utilização de estratégias comuns às duas áreas. As camadas STN já se mostraram eficientes em vários problemas supervisionados, logo faz sentido tentar utiliza-las neste novo contexto. Além disso, o algoritmo adaptativo de seleção de amostras pode ajudar as STNs a alcançar uma maior eficiência desde a primeira iteração, pois serão treinadas apenas com exemplos de treinamento com alta confiança de similaridade entre os pares durante

todo o processo. A partir da utilização dos demais modelos apresentados na revisão, esses benefícios não seriam obtidos.

## 2.2 ATENÇÃO VISUAL

Os seres humanos não conseguem processar todas as informações visualizadas em paralelo, isto devido a gargalos de processamento em nosso mecanismo de percepção. De acordo com (WANG; TAX, 2016), a atenção é um mecanismo que ajuda os seres humanos na concentração de apenas uma parte da informação ao qual se encontra exposto e, ao mesmo tempo, percebe com menos intensidade outros estímulos. Esta situação é simulada na Figura 5 onde no segmento a) ao olhar um cenário apenas alguns pontos são percebidos e tomam a total atenção do visualizador da cena. Nestas regiões, é possível notar os detalhes como podemos observar nos segmentos b) e c), enquanto diversas outras regiões do cenário não são devidamente percebidas pelo sistema cognitivo.

Figura 5 – Demostração dos gargalos de percepção visual humana. Olhando para um cenário os seres humanos conseguem concentrar sua atenção apenas em alguns pontos de interesse na imagem enquanto outros detalhes do cenário não são claramente percebidos.




Fonte: (KIMURA, 2015)

Motivados pelo mecanismo humano, a atenção visual é uma área em visão computacional. A atenção é aplicada para diminuir a área de busca do objeto de interesse na imagem, proporcionando ao modelo a capacidade de focar em regiões de interesse na imagem de

entrada, as quais são mais importantes para a execução da tarefa a qual se propõe resolver. Com isso, os mecanismos de atenção podem economizar recursos computacionais na realização de uma tarefa e otimizar sua capacidade de decisão. Existem duas categorias de abordagens, no desenvolvimento de estratégias de atenção em visão computacional. Estas são definidas como abordagens *bottom-up* e *top-down*(SUN; FISHER, 2003).

As abordagens *bottom-up* processam informações das imagens como cor, orientação, movimento, profundidade e outras características a um nível mais baixo de informação, para atrair a atenção. Um dos exemplos deste tipo de abordagem é a detecção de saliência, técnica que é amplamente utilizada em diversos trabalhos (CHENG et al., 2015). Esta técnica procura fortes características que possam realçar uma região em relação a outra. Por trabalhar com *features* de baixo nível, as abordagens *bottom-up* não conseguem capturar de forma eficiente informações semânticas relacionadas aos objetos da imagem(WANG; TAX, 2016).

Os trabalhos no paradigma *top-down* são guiados por informações prévias, contexto entre os elementos da imagem e objetivos. Uma das áreas mais marcantes do paradigma *top-down* é a busca visual (FRINTROP, 2011), que podem ser representadas por técnicas de janela deslizante. Estes métodos têm por objetivo deslizar uma janela de tamanho fixo sobre uma imagem, aplicando um processo de classificação em cada região para encontrar o objeto desejado. Sendo este outro modelo que combina com a ideia de atenção. Em muitos casos o objeto que se deseja encontrar na imagem localiza-se apenas em uma pequena região. Os métodos de janela deslizante são capazes de localizar estes objetos ao mesmo tempo em que diminuem o custo computacional, pois realiza a busca apenas na região delimitada pela janela. Entretanto as maiores áreas de aplicabilidade destes algoritmos de deslizamento de janelas são na detecção ou rastreamento de objetos em imagens digitais, sendo assim um mecanismo não universal e difícil de ser aplicado em diversas áreas em *machine learning* e visão computacional(WANG; TAX, 2016). As abordagens *top-down* representam uma área marcante e uma estratégia bem interessante a ser explorada em visão computacional(FRINTROP, 2011).

Atualmente, a quantidade de estudos que tem por objetivo utilizar atenção com a estratégia *top-dow* em modelos com arquiteturas *deep learning* tem aumentado. Neste cenário, são encontradas propostas bem sofisticadas, considerando problemas supervisionados. Como o *Recurrent Atention Model* (RAM) (MNIH et al., 2014), uma rede neural recorrente capaz de extrair informações de uma imagem ou vídeo por meio da seleção adaptativa de uma sequência de regiões ou locais, somente processando as regiões selecionadas das imagens de alta resolução. A seleção adaptativa é realizada com o auxílio de mecanismos chamados *glimpse sensor* que extraem alguns recortes em múltiplas resoluções da região contida na janela sobre a imagem. Este módulo envia esses recortes para a *glimpse network* construir as *features* utilizadas na *Recurrent Neural Network* (RNN), o processo auxilia na detecção da próxima janela a ser focada pelo método a partir da

utilização dos estados internos da rede. Além disso rede é treinada com uma estratégia de *reinforciment learning*.

O *Deep Recurrent Atention Model* (DRAM) (BA; MNIH; KAVUKCUOGLU, 2014) é uma evolução do método anterior e lida com a classificação e localização de múltiplos objetos em uma imagem. O *Enriched Deep Recurrent Atention Model* (EDRAM) (ABLAVATSKI; LU; CAI, 2017) é outro aprimoramento que combina DRAM com Camadas STN (JADERBERG et al., 2015) e torna a rede totalmente diferenciável podendo ser treinada com *Backpropagation*. Essa abordagem tem, em geral, a mesma estrutura apresentada na DRAM, mas usa o módulo STN modificado como um mecanismo de atenção para localizar a região da imagem de interesse e ajustar as transformações nos dados de entrada. Esta versão do módulo STN empregado no EDRAM pode usar informações de estados anteriores da rede recorrente para melhorar os resultados.

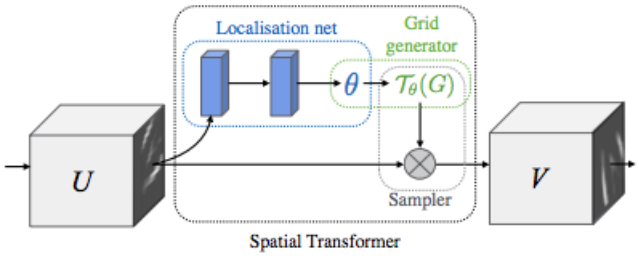
Como vimos, os modelos RAM e DRAM utilizam *reinforcement learning* para treinar suas redes por conter em sua arquitetura mecanismos não diferenciáveis. Sendo assim, novas ideias de mecanismos de atenção visual totalmente diferenciáveis que possam ser treinados em conjunto com a rede a qual estão acoplados foram propostas. Geralmente, estes mecanismos baseiam-se na ideia das camadas STN (JADERBERG et al., 2015) que são utilizadas em nosso modelo proposto e explicadas na próxima seção. Geralmente, estas modificações pretendem adicionar algumas propriedades ou funcionalidades às STNs. Como a *Deep Diffeomorphic Transformer Networks* (DETLEFSEN; FREIFELD; HAUBERG, 2018), com estrutura parecida com a STN, esta rede adicionalmente possibilita que a camada realize transformações difeomórficas. Este tipo de transformação pode ser invertida e evita perda de informação caso as STNs causem alguma distorção na imagem durante seu processo de otimização. Outra abordagem é a *Inverse Compositional Tranformer Networks* (LIN; LUCEY, 2016) que combina o algoritmo Lucas & Kanade com a camada STN. Esta abordagem permite que as camadas realizem as transformações no espaço de *features* com um maior alinhamento geométrico, potencializando as correções de eventuais transformações.

Diversas são as abordagens que utilizam atenção para melhorar os resultados em problemas de visão computacional, porém apenas algumas podem ser acopladas facilmente como um módulo em uma rede convolucional, como as camadas STN. Constatou-se que existem outras abordagens que tentam propor melhorias nesta camada desde sua concepção, porém ainda não foram testadas suficientemente em outros problemas além dos mostrados em seus respectivos trabalhos de origem. As STNs já são utilizadas em diversos segmentos, melhorando resultados em visão computacional em várias tarefas (CîRSTEA; LIKFORMAN-SULEM, 2016) (BEDNAREK; WALAS, 2018) (VU; HUANG, 2017). Devido a sua comprovada eficácia, as STNs foram escolhidas como o mecanismo de atenção para melhorar os resultados do modelo *Deep Adaptive Clustering*.

### 2.2.1 Spatial Transformer Network

A STN (JADERBERG et al., 2015) é um mecanismo de atenção visual que consiste em módulos diferenciáveis que podem ser treinados com o algoritmo *Backpropagation* e inseridos como um camada em redes convolucionais. Diferente das camadas *Max-Pooling* que aplicam transformações previamente fixadas nos dados de entrada, as STNs são mecanismos capazes de aprender e realizar transformações espaciais em apenas um *single-foward-pass* condicionadas ao mapa de dados de entrada. Desta forma estes módulos treinados realizam transformações nos dados de entrada que permitem a rede trabalhar apenas na área da imagem que contém o objeto de interesse para a realização da tarefa proposta. Estas camadas podem possuir múltiplos canais, que podem ser a imagem de entrada ou o mapa de características interno da rede. A mesma transformação aprendida é então aplicada da mesma forma em cada um dos canais. Mecanismos de atenção mais simples apenas permitem que o modelo foque em regiões de maior interesse na imagem. Além de possuir esta característica, as STNs são capazes de transpor os dados de entrada para um formato padrão com o intuito de facilitar o reconhecimento entre as demais camadas da rede. As *Spatial Transformer Networks* podem ser divididas em três partes distintas, como podemos ver na Figura 6, sendo estas a *localization network*, o *grid generator* e o *sampler*.

Figura 6 – *Spatial Transformer Network.*




Fonte: (CHANG et al., 2017)

Primeiramente, a camada recebe o mapa de dados de entrada $U \in \Re^{H \times W \times C}$ no qual $W$, $H$, $C$ são a largura, altura e canais, respectivamente. Essa pode ser a imagem de entrada ou o mapa de recursos extraídos pelas camadas internas de uma rede convolucional.

Este mapa é passado para a primeira parte da camada STN composta pela *localization network* $f_{\text{loc}}$, este módulo pode ser constituído por qualquer tipo de rede neural, sejam elas multicamadas tradicionais ou arquiteturas profundas, contanto que possuam uma camada de regressão na saída. Entende-se por camada de regressão uma camada densa, sem a adição de funções de ativação em sua saída. Logo, a partir do mapa de *features* ou imagens, sua tarefa principal é gerar o parâmetro $\theta$ para compor a transformação $T_\theta$. O tamanho de $\theta$ pode variar de acordo com o tipo de transformação selecionada. Podemos

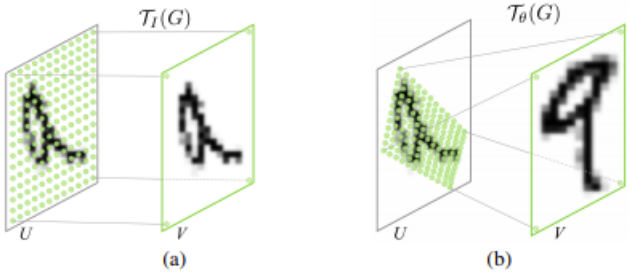
analisar o formato dos parâmetros de transformação e a *localization network* abaixo:

$$\begin{gathered} \theta = f_{\text{loc}}(U) \\ U \in \Re^{H \times W \times C} \end{gathered} \tag{2.17}$$

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \tau_\theta \begin{pmatrix} x_i^o \\ y_i^o \\ 1 \end{pmatrix} \tag{2.18}$$
$$\begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix}$$

Esta transformação não é aplicada diretamente sobre os mapas de *features*. Em seguida, para realizar a transformação no mapa de entrada, é gerada o *grid generator*, um grid de amostragem parametrizável disposto sobre a uma posição central da imagem original. Cada posição do *grid* corresponde a uma posição posteriormente gerada do mapa de saída. Sendo assim, os valores das coordenadas do mapa de entrada e do *grid* são normalizados para um intervalo entre [-1, 1]. A transformação $T_\theta$ é aplicada sobre o *grid*. Após esta etapa, é possível realizar a amostragem dos dados de entrada adequados para produzir a saída desejada, como mostrado na Figura 7. Em (a) é aplicado sobre o *grid* da transformação identidade $T_i$ e em (b) é realizada uma transformação $T_\theta$.

Figura 7 – a)Aplicação da transformação identidade no *sample grid*, b) aplicação de transformação $\theta$




Fonte: (JADERBERG et al., 2015)

Entretanto, as coordenadas transformadas não necessariamente correspondem a um número inteiro ou a um pixel exato do mapa de *features* de entrada, logo, para produzir o mapa de *features* de saída, é necessário a utilização de um *sampler* e acessar corretamente os valores do pixel de entrada. Segue abaixo a estrutura de um *sampler* genérico:

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c k\left(x_i^s - m; \Phi_x\right) k\left(y_i^s - n; \Phi_y\right) \forall i \in [1 \ldots H'W'] \quad \forall c \in [1 \ldots C] \tag{2.19}$$

Na definição do *sampler* exibida acima, $x_i^s, y_i^s$ definem as coordenadas em $T(\theta)$ que correspondem às localizações espaciais na entrada em que um *kernel* $k$ de amostragem é

aplicado para acessar um valor em pixel especifico de $V$. Ainda nesta equação, $\Phi_x$ e $\Phi_y$ são os valores genéricos do *kernel* de amostragem que define a interpolação da imagem, $U_{nm}^c$ é o valor da posição $(n,m)$ no canal $c$ do mapa de entrada. $V_i^c$ é o valor de saída do pixel $i$ na posição $(x_i^t, y_i^t)$ no canal $c$.

Através destes mecanismos, as redes convolucionais podem se tornar mais robustas e invariantes às transformações ou a variabilidade inerentes ao conjunto de dados das imagens de entrada e com baixo custo computacional. Em diversos trabalhos supervisionados podemos ver a eficiência e aplicabilidade das redes transformadoras.

Em (HALOI, 2015), por exemplo, as STNs são utilizadas em conjunto com uma versão modificada da rede *GoogleLenet* com camada *Inception* para classificação de sinais de trânsito alemães da base de dados GTSRB. Quatro STNs foram inseridas nas camadas iniciais do modelo, tornando o modelo mais robusto e invariante à transformações espaciais. Esta alteração permite que a rede classifique precisamente amostras intraclasse mesmo sobre deformações típicas de imagens capturadas no tráfego de automóveis.

Em (BEDNAREK; WALAS, 2018) uma camada STN é acoplada à entrada da rede convolucional para melhorar os resultados de classificação de dígitos manuscritos. Novamente no reconhecimento de dígitos manuscritos (CîRSTEA; LIKFORMAN-SULEM, 2016) utiliza STNs em conjunto a redes convolucionais que compartilham os mesmos pesos com a *localization network*. As STNs também são utilizadas para melhorar resultados na tarefa de detecção como vemos (VU; HUANG, 2017), onde é proposto um *framework* de detecção de vagas de estacionamento. As STNs ajudam a corrigir as transformações nas pequenas regiões da imagem onde possivelmente os carros estão estacionados.

Motivados por estes trabalho, nossa hipótese de pesquisa é que por meio das camadas STN temos uma grande chance de tornar a rede convolucional presente no modelo DAC mais robusta a alterações espaciais nas imagens de entrada e assim melhorar o resultado de *deep image clustering* atingidos pelo modelo base.

# 3 PROPOSTA: *SPATIAL TRANSFORMER - DEEP ADAPTIVE CLUSTERING* (ST-DAC)

O método proposto neste trabalho tem como objetivo criar um novo modelo adicionando ao DAC camadas STN para *deep image clustering*. Chamamos este novo modelo de *Spatial Transformer Deep Adaptive Clustering* (ST-DAC).

Nossa hipótese de pesquisa assume que a adição das camadas STN no modelo convolucional extrator de *features* componente do DAC irá tornar a rede mais eficiente ao lidar com as transformações existentes no conjunto de imagens de entrada. A ação das STNs não só nos dados de entrada, mas também nas *features* extraídas nas camadas internas irá permitir a rede realizar transformações nos dados para normalizar a forma dos objetos tratados. Espera-se que elas conduzam os elementos encontrados para uma área ou região em comum dentro do espaço de dimensões das *features*. Esta ação pode ainda diminuir a variância intraclasse e aumentar a similaridade entre os elementos do conjunto de dados que passam pelo processo de *clustering*.

Além disso, a própria forma como o DAC realiza seu processo de *clustering* pode favorecer um bom funcionamento das redes STN em sua arquitetura. A maneira como o modelo trata o problema de *clustering* encarando-o como uma tarefa de classificação de pares nos leva a crer que os bons resultados, antes obtidos em tarefas supervisionadas com o auxílio das STNs em rede convolucionais, possam ser repetidos com a sua utilização em tarefas não supervisionadas como o *clustering* de imagens. Outro ponto importante é que o método adaptativo de seleção de amostras também pode favorecer o treinamento das STNs, visto que desde as primeiras iterações são selecionadas para treinamento apenas rótulos gerados a partir das *features* extraídas pela rede convolucional do DAC, com um alto nível de confiança, evitando assim ruídos no processo de aprendizagem de ambos componentes, o DAC e as STNs.

Apesar de todos os indícios favorecerem a escolha de ambas as estratégias para compor o modelo proposto, as STNs não foram utilizadas até o momento para melhorar tarefas de *clustering* de imagens (YANG et al., 2016) (YANG; PARIKH; BATRA, 2016) (KINGMA; WELLING, 2013) (GOODFELLOW et al., 2014) (XIE; GIRSHICK; FARHADI, 2016) (GUO et al., 2018)(MIN et al., 2018) (ALJALBOUT et al., 2018). Sendo assim, o desafio principal é fazer com que essas duas arquiteturas funcionem em conjunto. Para isso, algumas modificações nas configurações originais de cada estratégia foram feitas para que os dois componentes DAC e STN funcionem em sintonia.

A arquitetura original de ambas estratégias foi analisada. Existem vários pontos de configuração nos métodos DAC e STN que podem ser considerados. No DAC, podem ser alterados, sem comprometer seu funcionamento, como proposto originalmente, os seguimentos a seguir:

- A rede convolucional utilizada para extrair as representações latentes da imagem;
- O otimizador utilizado na rede;
- Os parâmetros de configuração, como a taxa de aprendizado;
- Os limite inferior $L(\lambda)$ e superior $U(\lambda)$ de seleção de amostras.

No caso das STNs, pode ser alterada facilmente a rede profunda que gera o parâmetro $\theta$ responsável por criar a matriz de transformação.

Inicialmente, foram realizados experimentos utilizando as redes e configurações originais de cada técnica, a fim de compor o ambiente experimental e avaliar se, de fato, seria viável mantê-las. Visto que, originalmente, as configurações das STNs foram utilizadas em experimentos de classificação supervisionada (CîRSTEA; LIKFORMAN-SULEM, 2016) (BEDNAREK; WALAS, 2018) (VU; HUANG, 2017) (HALOI, 2015) e não de *clustering*, como pretendemos.

Foram pesquisadas as configurações e implementações do DAC no repositório do autor disponível em (CHANG et al., 2017), entretanto não encontramos a implementação do modelo DAC como originalmente proposto. No repositório, havia apenas disponível a sua versão modificada do método de *clustering Deep Adaptive Clustering*, uma versão chamada DAC* e também apresentada no artigo original (CHANG et al., 2017).

No DAC*, os limites superior $U(\lambda)$ e inferior de seleção $L(\lambda)$ de amostras são alterados pelo parâmetro $\lambda$, que nesta versão, não é aprendido pela rede. No modelo, este parâmetro começa com o mesmo valor do limite superior $U(\lambda)$ e segue sendo decrementado linearmente a cada época de treinamento até atingir o limite de seleção inferior $L(\lambda)$. Neste método, todos os exemplos são selecionados para treinamento em cada iteração, diferente do DAC que rejeita para treinamento os rótulos gerados por pares entre os limiares de seleção. Ainda assim, o método é capaz de selecionar rótulos pares para treino com alto nível de confiança aplicando os limiares.

Este modelo foi utilizado para compor a proposta deste trabalho, pois apesar de apresentar uma pequena variação nos resultados, em relação ao DAC, eles ainda superam os resultados de diversos outros métodos da literatura nos mesmos *benchmarks*. Além disso, ele possui a mesma estrutura do DAC e trata a tarefa de *clustering* como um problema de classificação binária, o que o torna adequado para a utilização de mecanismos de atenção no formato de camadas diferenciais como as STNs. Outro ponto de análise é se as STNs são capazes de recuperar esta pequena queda de desempenho em relação ao DAC, visto que as STNs já se mostraram eficazes em melhorar do desempenho de redes neurais em problemas de classificação supervisionada, inclusive com demonstrações de efetividade no artigo original da solução (JADERBERG et al., 2015).

A rede profunda convolucional, extratora de *features* latentes, que compõe a arquitetura presente no artigo original do DAC é a All-ConvNets (SPRINGENBERG et al., 2014).

Muito utilizada na classificação de imagens. Seu grande diferencial é que não aplica *max-pooling* entre a maioria das camadas convolucionais, optando por realizar pequenos *strides* entre elas. Essa estratégia difere bastante de arquiteturas mais clássicas como VGG(SIMONYAN; ZISSERMAN, 2014) ou LeNet (LECUN et al., 1998b). Com isso, a mesma obtém uma maior variabilidade de resoluções de representações de características tornando a rede mais robusta a transformações inerentes aos dados de entrada comparadas às obtidas na utilização das camadas de *max-pooling*. Este modelo, na tarefa de classificação, obteve bons resultados em diversas bases de dados e se mostra muito eficiente compondo o DAC como proposto originalmente em (CHANG et al., 2017). Decidimos então mantê-lo nos experimentos iniciais.

Na Tabela 1, podem ser observadas as configurações do DAC* utilizadas nos experimentos iniciais na tarefa de *image clustering*.

Tabela 1 – Configuração padrão do DAC*

| **DAC** |
|---|
| Rede Convolucional: All-ConvNets (SPRINGENBERG et al., 2014) |
| Otimizador: RMSprop (TIELEMAN; HINTON, 2012) |
| taxa de aprendizado: 0.001 |

Como sabemos, as STNs possuem uma *localization network*. Em (JADERBERG et al., 2015) se recomenda a utilização de uma rede com uma camada densa de regressão de saída, ou seja, sem posteriores funções de ativação. Este mesmo trabalho exibe a estrutura da rede padrão utilizada em seus experimentos e tem sua arquitetura apresentada na Tabela 2.

Tabela 2 – Arquitetura padrão das *localization networks* das STNs em (JADERBERG et al., 2015)

| ***Localization Network*(JADERBERG et al., 2015)** |
|---|
| 3 x 3 conv. 20 BN ReLU |
| 3 x 3 conv. 20 BN ReLU |
| 2 x 2 Maxpooling |
| 3 x 3 conv. 20 BN ReLU |
| 3 x 3 conv. 20 BN ReLU |
| 32 dense ReLU |
| 6 dense |

Inicialmente, propomos a inserção de uma camada STN antes da entrada da rede convolucional *All-ConvNets*, para avaliar o funcionamento da configuração como sugerido

em (JADERBERG et al., 2015). Executamos a rede cinco vezes na base de dados *MNIST* (LECUN et al., 1998a)[1] para a avaliação do resultado.

No entanto, nestes vários experimentos iniciais, tivemos dificuldade em treinar o modelo usando as camadas STN nesta configuração. Nesses casos, as camadas STN realizaram transformações que degradam as imagens, reduzindo a escala do objeto, fazendo-o sumir da área da imagem após várias iterações, ou até mesmo tornando a imagem de entrada ruidosa ou distorcida após algum tempo de treinamento e prejudicando o *clustering* das imagens. Estes experimentos acabaram atingindo resultados muito inferiores aos esperados, os valores de avaliação de acerto da rede encontravam-se em uma faixa entre 22% à 42%, resultados bem abaixo do que o método DAC sem as camadas STN já havia obtido na mesma base, os quais estão na ordem de 97.7%.

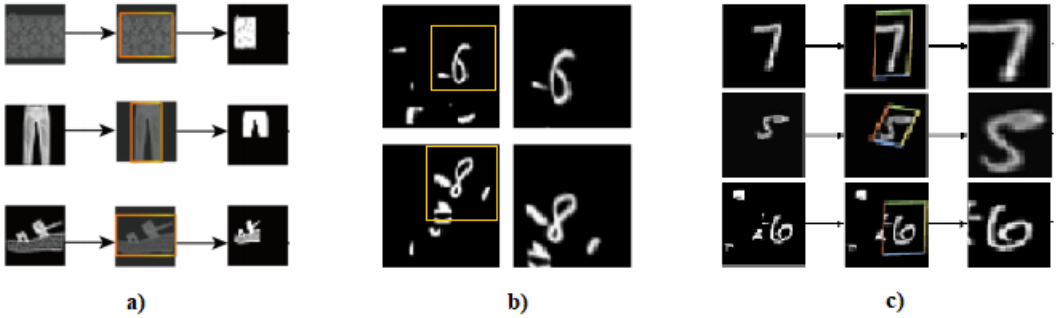
Esse comportamento nos levou a investigar se o problema poderia ser ocasionado devido ao *vanishing gradient problem* motivado pela profundidade da All-ConvNets ou a utilização de uma taxa de aprendizado que não atenda à tarefa em questão. Experiências semelhantes a estes problemas puderam ser encontradas na literatura. Como podemos ver em (MEI; GUO; YIN, 2018), os experimentos são conduzidos para realizar a classificação apenas na base de imagens *Fashion MNIST* (XIAO; RASUL; VOLLGRAF, 2017) utilizando redes convolucionais com o auxílio das camadas STN, o modelo utilizado para a extração de *features* é uma rede convolucional muito profunda assim como a *localization network* das camadas STN. Os resultados de classificação exibidos no artigo tiveram algum ganho com a utilização das STNs, entretanto as transformações realizadas pelas mesmas e exibidas no trabalho em questão, não correspondem ao comportamento normal das STNs, apresentado em outros trabalhos na literatura (CîRSTEA; LIKFORMAN-SULEM, 2016) (VU; HUANG, 2017) (JADERBERG et al., 2015), nos quais as STNs corrigiam distorções e aplicavam um forte *zoom* de aproximação nos objetos.

Assim como em nosso caso, em (MEI; GUO; YIN, 2018) os objetos das imagens sofreram uma redução de escala, muito provavelmente por problemas de treinamento da rede. Podemos verificar os exemplos de transformações exibidos na Figura 8. No segmento a) podemos observar as transformações que as STNs causaram nas imagens uma redução do objeto de interesse, não focando em suas principais características. No segmento b) e c) podemos ver o resultado das camadas STN nos trabalhos de (CîRSTEA; LIKFORMAN-SULEM, 2016) e (JADERBERG et al., 2015), respectivamente. Nestes casos, o comportamento se mostra diferente, corrigindo as distorções nas imagens e aproximando o objeto de interesse, o que é de fato o esperado e o cenário mais encontrado na literatura.

No trabalho (MEI; GUO; YIN, 2018), a validação é apenas realizada em uma base de imagens; talvez se avaliada a eficiência do modelo em outras bases de dados, ou se os autores tivessem executado o método por mais iterações, detectariam o problema e os resultados do modelo seriam inferiores aos valores do *benchmark* como no nosso caso. Sendo

[1] Detalhes sobre esta base de dados são apresentados no capítulo de experimentos

Figura 8 – a) Redução de escala do objeto de interesse causado pelas STNs em (MEI; GUO; YIN, 2018) semelhante ao cenário encontrado em nossos experimentos iniciais. b) e c) cenários esperados da utilização das STNs apresentados em (CiRSTEA; LIKFORMAN-SULEM, 2016)(JADERBERG et al., 2015) respectivamente. As *localizations networks* realizam transformações para corrigir distorções e focar no objeto de interesse.



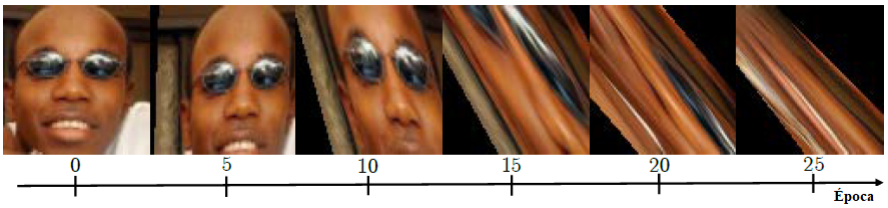

Fonte: Adaptada de (MEI; GUO; YIN, 2018) (CiRSTEA; LIKFORMAN-SULEM, 2016) (JADERBERG et al., 2015)

as redes utilizadas neste trabalho muito profundas, acreditamos que os sinais propagados até as STNs não eram suficientes para um bom treinamento das mesmas.

Levamos em consideração que outra possível causa das distorções que ocorriam nas imagens poderia ter sido ocasionada pela utilização de uma taxa de aprendizado inadequada que não atenda ao treinamento da rede convolucional do DAC* e das redes localizadoras das STNs ao mesmo tempo.

Em (DETLEFSEN; FREIFELD; HAUBERG, 2018), encontramos uma explicação sobre o funcionamento das STNs; nele é afirmado que as STNs são sensíveis à taxa de aprendizado e retornam transformações afins destrutivas depois de um certo período de treinamento, caso a taxa de aprendizado seja inadequada, como podemos acompanhar na Figura 9. Estas distorções também se assemelham bastante com as obtidas em nossos experimentos iniciais. Neste mesmo trabalho, é comprovado que para uma boa utilização das STNs a taxa de aprendizado utilizada nos treinamentos dos modelos deve ter valores próximos a 0,0001. Logo, passamos a utilizar essa taxa em nossos experimentos.

Figura 9 – Distorções causadas pela STN quando utilizada uma taxa de aprendizado inadequada.




Fonte: Adaptada de (DETLEFSEN; FREIFELD; HAUBERG, 2018)

Para minimizar o problema de distorções e distanciamento possivelmente causados também pelo *vanishing gradient problem*, optamos por substituir a rede padrão por um modelo menor, baseado na rede VGG (SIMONYAN; ZISSERMAN, 2014); esta rede também possui bons resultados em várias bases de dados da literatura. O modelo VGG, como

proposto originalmente, possui muitas camadas, assim como a All-ConvNets. Para que o mesmo cenário problemático não ocorresse novamente, decidimos diminuir o número de camadas da arquitetura, para uma quantidade onde seja possível o bom funcionamento entre a estratégia DAC* e as camadas STN.

Realizamos experimentos para avaliar a alteração da rede no modelo DAC*. A substituição da rede All-ConvNets pela rede baseada em VGG não alterou significativamente o desempenho do modelo original, como é apresentado no capítulo de resultados. Este cenário nos ofereceu a oportunidade de avaliar, ainda mais, a capacidade das STNs em melhorar algoritmos de *clustering*, caso consiga melhorar o desempenho dos resultados do método DAC em relação à configuração original. A arquitetura da rede convolucional baseada em VGG pode ser vista na Tabela 3.

Tabela 3 – A arquitetura da rede convolucional utilizada em nossos experimentos.

| **Modelo de rede convolucional baseado em VGG** |
|---|
| Input 28x28 imagem monocromática |
| 3 x 3 conv. 64 BN ReLU |
| 2 x 2 Maxpooling BN |
| 3 x 3 conv. 128 BN ReLU |
| 2 x 2 Maxpooling BN |
| 3 x 3 conv. 256 BN ReLU |
| 2 x 2 Maxpooling BN |
| 3096 dense BN ReLU |
| 10 dense BN ReLU SoftMax |
| Em alguns experimentos as camadas STN são removidas. |

No modelo proposto trocamos também o otimizador RMSprop(TIELEMAN; HINTON, 2012) originalmente usado no DAC pelo Adam(KINGMA; BA, 2014) para acelerar a convergência da rede. Além disso, nossa escolha foi motivada pela utilização deste método no treinamento de redes convolucionais com STN em outros trabalhos (DETLEFSEN; FREIFELD; HAUBERG, 2018) (CîRSTEA; LIKFORMAN-SULEM, 2016). Na Tabela 4 são apresentadas as configurações iniciais do nosso modelo proposto ST-DAC.

Inserimos diferentes combinações de camadas STN, na camada de entrada e também após os filtros convolucionais, no total usamos 3 camadas. A Figura 10 apresenta a arquitetura final do modelo proposto com as camadas STN. Inserimos uma camada STN próxima a cada bloco de camadas convolucionais, aplicando as correções afins aos dados de entrada originais, como recomendado em (JADERBERG et al., 2015). As STNs aplicam, desta forma, transformações nas características extraídas pela camada convolucional, a cada vez que as imagens sofrem uma transformação espacial significativa, como após uma

Tabela 4 – Configuração propostas

| **DAC*** |
|---|
| Rede Convolucional: Baseada em VGG(SIMONYAN; ZISSERMAN, 2014) |
| Otimizador: Adam |
| taxa de aprendizado: 0,0001 |

redução de resolução pela camada de *max-pooling*. Com essa estrutura, esperamos corrigir a transformação nos dados realizada com a alteração na resolução da representação.

O ST-DAC não utiliza uma camada STN após o primeiro bloco de camadas convolucionais, para reduzir o custo computacional, pois as correções já são feitas previamente na mesma resolução nos dados de entrada. A estrutura da rede de localização nas camadas espaciais é a mesma proposta em (DETLEFSEN; FREIFELD; HAUBERG, 2018) e é detalhada na Tabela 5.

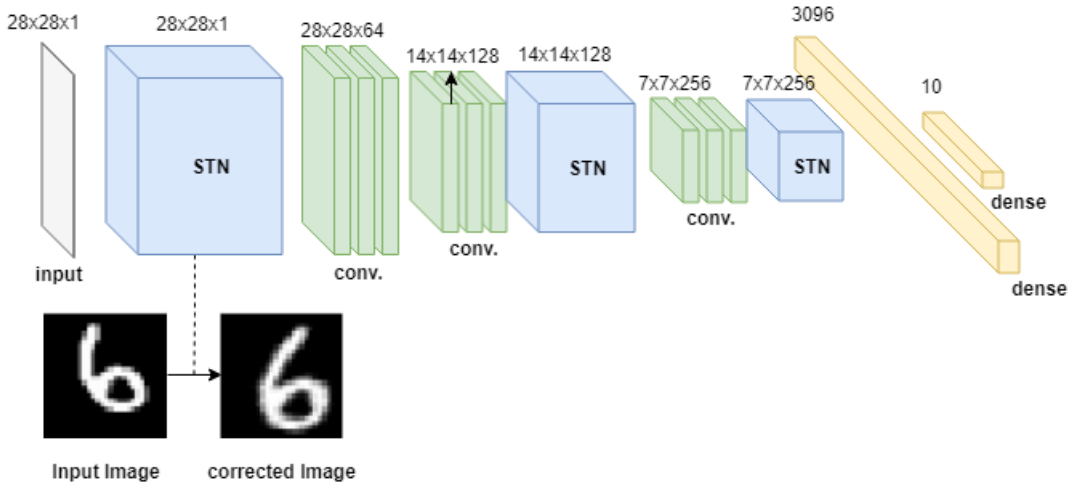
Tabela 5 – A arquitetura da *Localization Network* utilizada do ST-DAC

| **Localization Network ST-DAC** |
|---|
| Input NxNxM imagem monocromática |
| 2 x 2 Maxpooling |
| 5 x 5 conv. 20 Tanh |
| 2 x 2 Maxpooling |
| 5 x 5 conv. 20 Tanh |
| 50 dense Tanh |
| 6 dense Tanh |
| Na camada de entrada NxNxM corresponde às dimensões de saída da camada anterior ao módulo STN. |

Normalmente, as estruturas de redes de localização usam a função de ativação *Rectified Linear Unit* (ReLU) entre as camadas e deixam a última camada sem ativação (JADERBERG et al., 2015). De uma maneira não convencional, (DETLEFSEN; FREIFELD; HAUBERG, 2018) usa funções de ativação do tipo tangente hiperbólica após todas as camadas convolucionais e densas. Nos experimentos iniciais, esta configuração apresentou melhores resultados com as camadas STN do que as estruturas convencionais com ReLU.

O modelo proposto ST-DAC, consegue realizar o processo de *deep image clustering* de forma mais eficiente, seguindo as configurações encontradas na literatura o funcionamento das STNs ocorre de forma esperada. Em seguida comprovamos seu funcionamento nas avaliações por meio de experimentos e análise dos resultados.

Figura 10 – A arquitetura convolucional proposta no ST-DAC possui três camadas transformadoras espaciais. A primeira é inserida após a camada de entrada e realiza transformações na imagem inicial. As outras camadas espaciais são aplicadas nos mapas de características após o segundo e terceiro bloco de camadas convolucionais.




Fonte: o autor

# 4 METODOLOGIA E EXPERIMENTOS

## 4.1 BASES DE DADOS

Os experimentos foram realizados com duas bases de dados amplamente utilizadas na validação de modelos de visão computacional: a base de dados de dígitos manuscritos MNIST(LECUN et al., 1998a) e a base composta por imagens de vestuário Fashion MNIST(XIAO; RASUL; VOLLGRAF, 2017).

### 4.1.1 MNIST

A base de dados MNIST é muito utilizada para avaliar problemas de aprendizado de máquina e visão computacional, diversos trabalhos de *deep image clustering* a utilizam para avaliar seus métodos como(YANG; PARIKH; BATRA, 2016)(GOODFELLOW et al., 2014)(XIE; GIRSHICK; FARHADI, 2016)(GUO et al., 2018). Este conjunto formado por 70.000 imagens monocromáticas de dígitos manuscritos. É composto por 60.000 imagens para treinamento e 10.000 para testar o modelo, todas as imagens têm um tamanho de 28x28 pixels com elementos pertencentes a 10 classes de dígitos de 0 a 9, alguns exemplos de cada classe podem ser vistos na Figura 11.

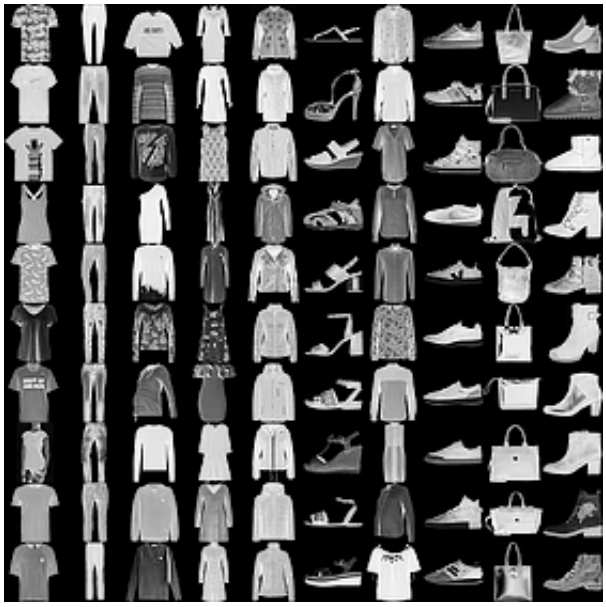
Figura 11 – Base de dados MNIST, composta por imagens de dígitos manuscritos pertencentes a 10 diferentes classes.

Fonte: o autor

### 4.1.2 Fashion MNIST

O segundo conjunto de dados utilizado é o Fashion MNIST, desenvolvido para disponibilizar um desafio maior que o MNIST e possuir o mesmo formato das imagens do MNIST. O conjunto é formado por 70.000 imagens monocromáticas de peças de vestuário, composto por 60.000 imagens para treino e 10.000 imagens de teste. As imagens possuem um tamanho de 28x28 pixels com elementos pertencentes a 10 classes; exemplos de cada classe podem ser visualizados na Figura 12.

Figura 12 – Base de dados Fashion MNIST, composta por imagens de vestuário pertencentes a 10 diferentes classes.



Fonte: o autor

## 4.2 MEDIDAS DE AVALIAÇÃO

As medidas utilizadas para avaliar os modelos foram a *Adjusted Rand Index* (ARI), *Normalized Mutual Information* (NMI), e *Clustering Accuracy* (ACC). Estas medidas retornam resultados em um intervalo entre [0,1]; valores próximos de 1 representam resultados de *clustering* mais precisos. Vários trabalhos utilizam estas meddidas para avaliação dos modelos de *deep image clustering*(CHANG et al., 2017)(GUO et al., 2018)(YANG; PARIKH; BATRA, 2016).

### 4.2.1 *Clustering Accuracy* (ACC)

O ACC é calculado da seguinte forma: para a imagem $i_{th}$, vamos denotar $q_i$ como o resultado do algoritmo de *clustering* e $p_i$ como o rótulo de *ground truth*. O $ACC$ é definido

como:

$$\text{ACC} = \frac{\sum_{i=1}^{n} \delta\left(p_i, \text{map}\left(q_i\right)\right)}{n} \tag{4.1}$$

$n$ é o número total de imagens presentes no conjunto de dados, $\delta$ é uma função onde $\delta(x, y) = 1$, se $x = y$ ($\delta(x, y) = 0$, em outro caso), e $map(q_i)$ é uma função de mapeamento que realiza a correspondência entre os valores atribuídos pelo método de *clustering* a cada elemento e o seu rótulo de validação. Obtivemos esta correspondência utilizando o algoritmo Kuhn-Munkres (KUHN, 2005), como em (CHANG et al., 2017).

### 4.2.2 *Normalized Mutual Information* (NMI)

O NMI calcula a taxa de informação que podemos saber sobre as classes $U$ dado os *clusters* $V$ para a informação normalizada que é contida em $U$ e $V$.

$$\text{NMI} = \frac{\text{MI}(U, V)}{\text{média}(H(U), H(V))} \tag{4.2}$$

A entropia $H$ é a quantidade de incerteza de uma partição, $P$ é a probabilidade de um objeto selecionado aleatoriamente a partir de $U$ ser rotulado na classe $U_{ith}$. Analogamente, $P'$ é a probabilidade de um objeto selecionado aleatoriamente a partir $V$ ser atribuído ao *cluster* $V_{ith}$ . $P(i, j)$ é a probabilidade que um objeto escolhido aleatoriamente esteja nas duas classes $U$ e $V$. A informação mútua é definida na equação abaixo:

$$\text{MI}(U, V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} P(i, j) \log\left(\frac{P(i, j)}{P(i)P'(j)}\right) \tag{4.3}$$

A informação mútua normalizada é definida como:

$$\text{NMI} = \frac{\sum_{i=1}^{|U|} \sum_{j=1}^{V} \frac{|U_i \cap V_j|}{N} \log\left(\frac{N|U_i \cap V_j|}{|U_i||V_j|}\right)}{\text{mean}(-\sum_{i=1}^{|U|} P(i)\log(P(i)), -\sum_{i=1}^{|V|} P'(i)\log(P'(i)))} \tag{4.4}$$

### 4.2.3 *Adjusted Rand Index* (ARI)

O ARI é recomendado para medir a concordância, mesmo quando as partições comparadas têm diferentes números de grupos. Sendo assim, ARI é uma função que mede a similaridade das duas atribuições, ignorando as permutações e com resultados superiores de normalização. Se $C$ é o rótulo da classe do objeto e $K$ é o valor de atribuição de classe do *cluster*, definimos $a$ como o número de pares de elemento que estão no mesmo conjunto em $C$ e ao mesmo tempo no grupo $K$, e $b$ o número de pares de elementos que estão em conjuntos em $C$ e em conjuntos diferentes em $K$.

O *rand index* (não-ajustado) pode ser obtido a partir de:

$$\text{RI} = \frac{a + b}{C_2^{n\ \text{samples}}} \tag{4.5}$$

Entretanto o valor $RI$ não garante que a atribuição de rótulos randômicos irá obter um valor perto de zero, especialmente se o número de grupos está na mesma ordem de magnitude que o número de amostras.

Para contornar este efeito, podemos descontar o $RI$ esperado das rotulagens aleatórias, definindo o ARI, da seguinte forma:

$$\text{ARI} = \frac{\text{RI} - E[\text{RI}]}{\max(\text{RI}) - E[\text{RI}]} \tag{4.6}$$

### 4.2.4 Configurações experimentais

Para avaliar o funcionamento e desempenho do modelo proposto ST-DAC, realizamos experimentos comparando a abordagem proposta com o DEC (XIE; GIRSHICK; FARHADI, 2016), DCN (YANG et al., 2016), VADE (JIANG et al., 2017), JULE (YANG; PARIKH; BATRA, 2016); dentro do trabalho DEC-DA (GUO et al., 2018), a abordagem que mais se destaca é a ConvDEC-DA. Desta forma, também comparamos nossos resultados com este método. Além destes, comparamos o modelo proposto com o DAC (CHANG et al., 2017), e sua versão DAC* original. Os modelos utilizados para comparação apresentam os melhores resultados de literatura nas duas bases de dados avaliadas.

Para verificar como o uso das camadas STN pode contribuir para o resultado final do modelo, comparamos nossa abordagem em quatro experimentos:

- Com todas as camadas STN ativadas (ST-DAC + 3 camadas STN) ;
- Com a última camada desligada (ST-DAC + 2 camadas STN);
- Com a primeira camada ativada (ST-DAC + 1 camada STN);
- Sem camadas STN (ST-DAC sem camada STN).

A ideia é quantificar a contribuição das camadas STN na precisão do modelo proposto ST-DAC.

O artigo original do DAC (CHANG et al., 2017) não realiza experimentos no banco de dados do Fashion MNIST. Para fins de comparação, a versão do DAC* foi executada nesta base de dados, usando os mesmos parâmetros sugeridos no documento original (CHANG et al., 2017). Adicionalmente, realizou-se a execução do modelo na base MNIST. Foi Utilizado para comparação o melhor resultado obtido por estes modelos nestes experimentos durante as épocas.

Usamos os mesmos parâmetros de *data augmentation* apresentados no artigo original do DAC (CHANG et al., 2017), em todos os experimentos com as duas bases de imagens. Modificamos no ST-DAC os limiares de seleção inicial inferior $L(\lambda)$ e superior $U(\lambda)$, respectivamente, para um intervalo entre [0,9 0,99] no experimento do conjunto de dados

MNIST e entre [0,8 0,99] para o conjunto de dados do Fashion MNIST. Utilizamos o otimizador Adam em nosso modelo com uma taxa de aprendizado de 0,0001, como sugerido em (DETLEFSEN; FREIFELD; HAUBERG, 2018) e (CHANG et al., 2017).

Todos os experimentos foram repetidos 10 vezes, nos quais foram calculados a média e o desvio padrão das medidas de avaliação ACC, NMI e ARI. Estas medidas são usadas para comparação com os outros métodos. Na Tabela 7 são apresentados os resultados dos métodos DAC, DAC*, DEC, DEC-DA, DCN, VADE e JULE, retirados de trabalhos na literatura, as fontes seguem próximas ao nome dos métodos. Os resultados marcados com - não se encontraram disponíveis. Os demais apresentados na Tabela 6 foram obtidos a partir de experimentos realizados por nós.

Além dos resultados obtidos nos experimentos, avaliamos o comportamento das STNs durante toda a etapa de treinamento com o intuito de avaliar a contribuição das mesmas na efetividade do modelo durante todo processo. Para isso, foi realizada a amostragem dos resultados obtidos pelo ST-DAC durante as épocas de treinamento e algumas imagens da primeira camada STN foram extraída do modelo para verificar a qualidade das transformações aplicadas na imagem de entrada durante todas as etapas de treinamento. Posteriormente, calculamos a imagem média, a variância e o desvio padrão de cada classe nas duas bases de dados. O objetivo da análise é comparar as saídas de cada agrupamento gerado pela primeira camada STN do modelo ST-DAC e também considerando as diferentes combinações de camadas STN no modelo, permitindo assim inspecionar a contribuição das camadas STN em cada agrupamento. Com estes artefatos, torna-se possível avaliar de que forma as STNs contribuíram no processo de aprendizado e se o algoritmo adaptativo inerente ao modelo auxiliou no funcionamento e sintonia das mesmas.

## 4.3 RESULTADOS E DISCUSSÃO

Os resultados dos experimentos com os métodos avaliados podem ser observados na Tabela 6. O modelo proposto, sem as camadas STN, apresentou resultados inferiores aos obtidos pelo modelo DAC. Este é um cenário esperado porque, comparado ao modelo DAC original, no ST-DAC usamos uma rede convolucional menos profunda para extrair as representações das imagens. No entanto, podemos observar que nossa abordagem ST-DAC usando camadas STN supera os resultados obtidos pelos outros métodos em todas as métricas na base *Fashion MNIST* (Tabela 7), atingindo o estado-da-arte. Além disso, na base MNIST o modelo tem um grande ganho em relação a sua versão sem camadas STN, superando igualmente o DAC, DAC*, JULE, VADE, DCN e DEC e se aproximando bastante do estado-da-arte, representado pelo desempenho do ConvDEC-DA. Uma ressalva que precisa ser feita é que como os experimentos não foram feitos com as mesmas condições, uma comparação direta não pode ser feita. Porém, o modelo ST-DAC treinado com e sem camadas STNs foi executado sobre os mesmos parâmetros de configuração, sendo

assim, sugere que o ganho em desempenho e resultados do modelo foram resultantes da utilização das camadas STN em sua rede convolucional.

Tabela 6 – Desempenho de *clustering* dos diferentes métodos sobre os conjuntos de imagem, a patir de experimentos executados para este trabalho, considerando *Clustering Accuracy* (ACC), *Normalized Mutual Information* (NMI) e *Adjusted Rand Index* (ARI).

| **Modelo** | **Base de Dados** | | | | | |
|---|---|---|---|---|---|---|
| | **MNIST** | | | **Fashion MNIST** | | |
| | ***ACC*** | ***NMI*** | ***ARI*** | ***ACC*** | ***NMI*** | ***ARI*** |
| DAC* | $0.974 \pm 0.004$ | $0.944 \pm 0.006$ | $0.945 \pm 0.008$ | $0.628 \pm 0.048$ | $0.589 \pm 0.034$ | $0.483 \pm 0.042$ |
| ST-DAC sem camadas STN | $0.957 \pm 0.038$ | $0.932 \pm 0.029$ | $0.923 \pm 0.051$ | $0.612 \pm 0.025$ | $0.591 \pm 0.024$ | $0.460 \pm 0.025$ |
| ST-DAC + 1 camada STN | $0.978 \pm 0.006$ | $0.950 \pm 0.008$ | $0.953 \pm 0.012$ | $0.649 \pm 0.035$ | $0.650 \pm 0.027$ | $0.520 \pm 0.032$ |
| ST-DAC + 2 camadas STN | $\mathbf{0.980 \pm 0.007}$ | $\mathbf{0.953 \pm 0.010}$ | $\mathbf{0.956 \pm 0.014}$ | $0.656 \pm 0.046$ | $0.658 \pm 0.032$ | $0.533 \pm 0.045$ |
| ST-DAC + 3 camadas STNs | $0.961 \pm 0.033$ | $0.936 \pm 0.022$ | $0.927 \pm 0.040$ | $\mathbf{0.664 \pm 0.044}$ | $\mathbf{0.668 \pm 0.025}$ | $\mathbf{0.541 \pm 0.044}$ |

Tabela 7 – Desempenho de *clustering* dos diferentes métodos sobre os conjuntos de imagem, retirados da literatura, considerando *Clustering Accuracy* (ACC), *Normalized Mutual Information* (NMI) e *Adjusted Rand Index* (ARI).

| **Modelo** | **Base de Dados** | | | | | |
|---|---|---|---|---|---|---|
| | **MNIST** | | | **Fashion MNIST** | | |
| | ***ACC*** | ***NMI*** | ***ARI*** | ***ACC*** | ***NMI*** | ***ARI*** |
| DCN(GUO et al., 2018) | 0.830 | 0.810 | - | 0.501 | 0.558 | - |
| VADE(GUO et al., 2018) | 0.945 | 0.876 | - | 0.578 | 0.630 | - |
| JULE(GUO et al., 2018) | 0.964 | 0.913 | - | 0.563 | 0.608 | - |
| DEC(GUO et al., 2018) | 0.863 | 0.834 | - | 0.518 | 0.546 | - |
| ConvDEC-DA(GUO et al., 2018) | **0.985** | **0.962** | - | **0.586** | **0.636** | - |
| DAC*(CHANG et al., 2017) | 0.966 | 0.924 | 0.940 | - | - | - |
| DAC(CHANG et al., 2017) | 0.977 | 0.935 | 0.948 | - | - | - |

Utilizando apenas uma camada STN após a camada de entrada do modelo proposto, obtém-se um resultado superior em quase todas as métricas nos experimentos com as duas bases de dados, comparado aos melhores resultados anteriormente obtidos pelo DCN, JULE, VADE, DEC, DAC* e DAC.

Adicionando uma camada STN antes da entrada e outra aplicada aos recursos extraídos após o segundo bloco de camadas convolucionais, fomos capazes de superar os melhores resultados anteriores do DCN, JULE, VADE, DEC, DAC* e DAC nos dois conjuntos de dados e superando o ConvDEC-DA e o DEC na Fashion MNIST por uma grande margem de acerto.

Comparando os resultados dos experimentos usando modelos com um número diferente de camadas STN, é notável que, mesmo com apenas uma camada, o modelo obtém uma melhora considerável, em comparação com sua versão original sem as camadas STN. Também é notável que, com a adição de mais camadas STN, o modelo pode obter melhores resultados. No entanto, a ideia de usar mais camadas para melhorar os resultados não pode ser aplicada em todos os contextos. Usando três camadas STN nos experimentos com a

base do Fashion MNIST, obtemos o melhor resultado neste conjunto de dados. No entanto, nos experimentos com o MNIST, o uso de uma terceira camada STN comprometeu de alguma forma os dados, reduzindo seu resultado final em comparação com a rede com duas camadas STN em execução neste mesmo conjunto de dados.

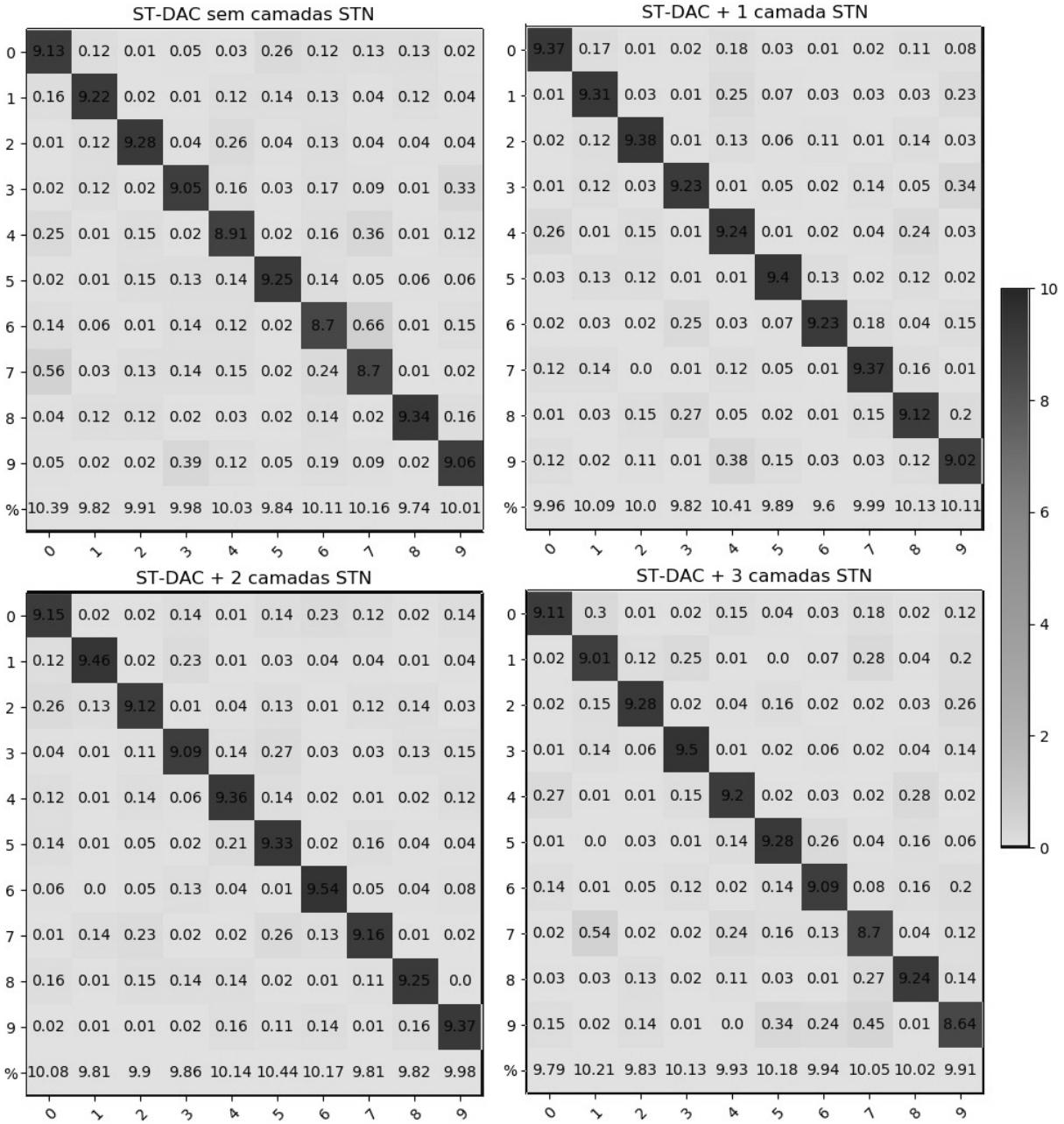
Para realizar uma discussão mais precisa sobre os dados obtidos, aprofundamos nossa análise construindo a matriz de confusão a partir dos resultados extraídos das variações propostas do modelo ST-DAC com diferentes números de camadas STN. Com este artefato podemos analisar a distribuição dos erros e acertos entre as classes das bases avaliadas de forma mais detalhada.

A matriz foi construída a partir da soma dos resultados de classificação das dez execuções dos experimentos. Os valores de cada célula foram calculados a partir da média dos resultados entre os experimentos, dividindo as somas de cada célula de classificação da matriz, pela quantidade de elementos da base completa, multiplicada pelo número de execuções realizadas. Com isto obtemos a taxa de classificação média (%) para cada célula baseada na quantidade de elementos existentes na base de dados. Na matriz, as 10 primeiras linhas correspondem a quantidade de elementos da classe existente na base de dados, sua soma corresponde a 10%, já que a quantidade de elementos das dez classes de cada base são distribuídos uniformemente, as colunas representam o *cluster* médio considerando cada classe formada entre as execuções. A última linha é uma informação adicional sobre a taxa média da quantidade de elementos de toda a base de dados distribuídos entre os *clusters* de cada classe, entre os experimentos. Cada célula da diagonal da matriz contém a taxa de acerto da classe em relação ao seu respectivo *cluster*, células desta diagonal com valores mais próximos de 10%, correspondem a um melhor resultado de clusterização.

Na Figura 13, é apresentada a matriz de confusão dos resultados dos modelos ST-DAC executados na base MNIST. Analisando a matriz do ST-DAC sem as camadas STN, percebemos como as transformações dos dígitos manuscritos, que são inerentes ao estilo de escrita presente em cada exemplo, confundem a rede, gerando falsos positivos até mesmo entre classes de dígitos que possuem características estruturais bem distintas. Como observamos no *cluster* 0 no qual existe uma quantidade acima da média de erro, relacionado à falsos positivos com o digito 7. Caso semelhante acontece com o *cluster* 7, com falsos positivos advindos das classes 6 e 4, assim como o *cluster* 3 com falsos positivos acima da média de erro com a classe 9. Estes exemplos de erro de agrupamento, que poderiam ser facilmente evitados com a normalização das transformações dos dígitos, reduzem a média geral de acertos do modelo.

Analisando os resultados do modelo ST-DAC + 1 camada STN, vemos que os erros apresentados no cenário anterior são reduzidos. As transformações realizadas pela rede conseguem ajudar a reduzir os falsos positivos dos elementos com características estruturais distintas, reduzindo os erros dos cenários previamente comentados. Isso acarreta em

Figura 13 – Matriz de confusão calculada a partir da taxa de acerto *de clustering*, dos modelos ST-DAC, na base MNIST




Fonte: o autor

um aumento do desempenho e a diagonal com a taxa de acerto de cada classe passa a conter apenas taxas acima dos 9%, ainda assim, a rede apresenta falsos positivos elevados entre classes de dígitos que possuem similaridades estruturais, como os dígitos 4 e 9 ou 3 e 9.

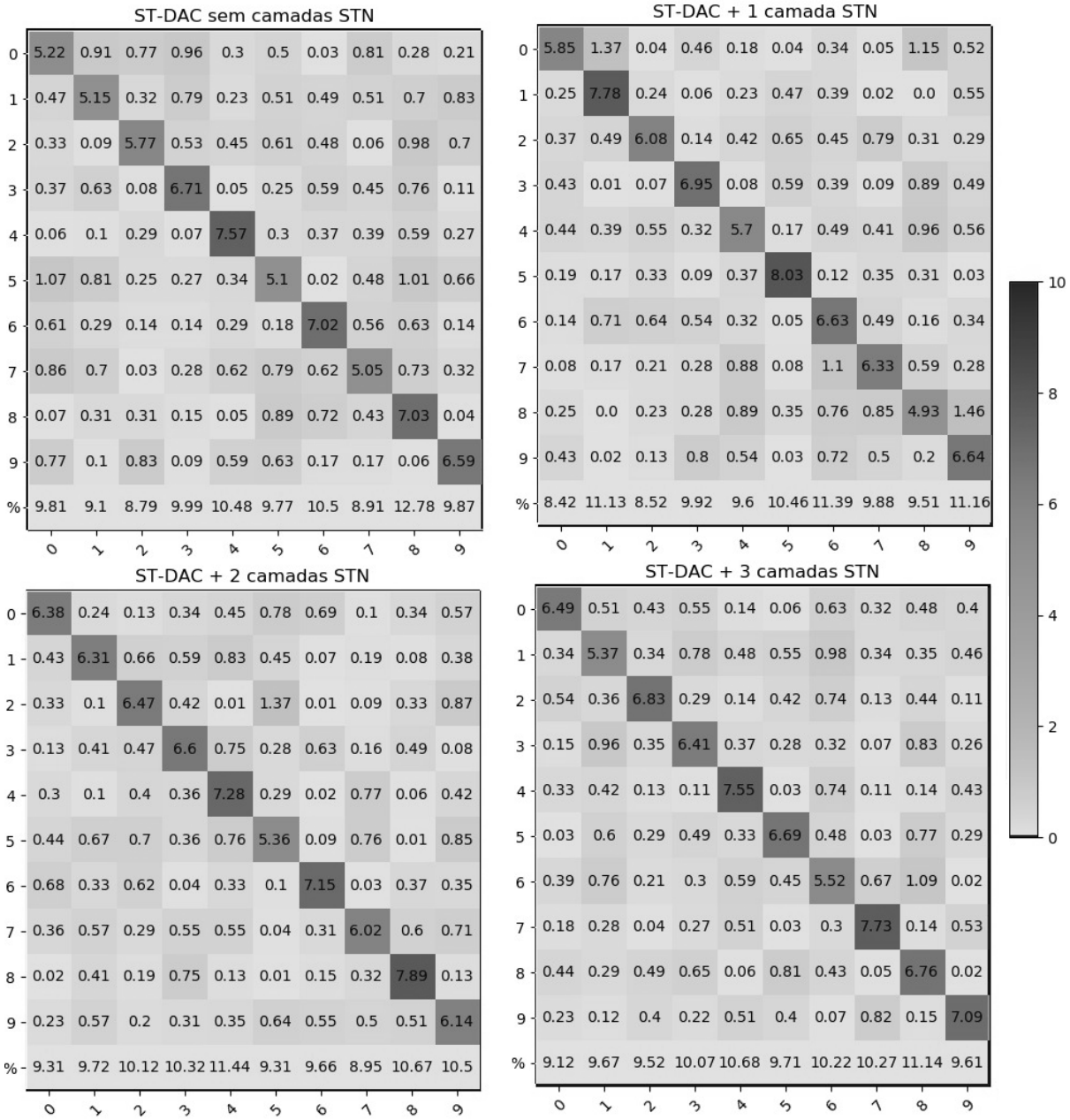
Adicionando mais uma camada STN é possível proporcionar mais robustez à rede e solucionar alguns destes problemas, como apresentado na matriz do ST-DAC + 2 camadas STNs. Nos resultados deste modelo os falsos positivos entre dígitos com estruturas semelhantes é reduzido consideravelmente, no modelo com a melhor taxa de acerto de *clustering* na base MNIST. Com a adição de uma nova camada os falsos positivos retornam, como vemos no modelo ST-DAC + 3 camadas STN, principalmente entre as classes com pequenas variações estruturais como os dígitos 1 e 7, assim como 5 e 9. Acreditamos que o mapa de *features* de resolução menor perde detalhes que distinguem classes distintas, porém, semelhantes em sua estrutura. Sendo assim, a rede da camada STN é confundida e aplica as mesmas transformações em comum às duas classes nesse espaço de *features*, gerando assim falsos positivos no modelo de *clustering* entre os dígitos das classes distintas.

A base Fashion MNIST representa um desafio maior do que o apresentado na base MNIST. Os elementos intraclasse em muitos casos apresentam uma grande variância no formato estrutural. Existindo uma divisão de estilos e estruturas entre diversas partições de elementos de um mesmo grupo e semelhanças extra grupo. Agrupar elementos de uma mesma categoria nesse contexto é uma tarefa mais difícil. A dificuldade também se reflete na análise dos resultados por classe, pois como a variância intraclasse é consideravelmente grande e muitas destas variações provavelmente não podem ser corrigidas por transformações afins, torna-se difícil a tarefa de encontrar os padrões de transformações aplicados a imagens específicas.

Visto este cenário, analisamos os resultados da rede ST-DAC sem camadas STN na base Fashion MNIST na Figura 14, podemos notar um cenário bem diferente do anterior, obtido com a base MNIST, a taxa de falsos positivos é bem maior, principalmente entre classes que possuem formatos semelhantes. Nesta base encontramos classes bem parecidas entre si, se considerarmos a estrutura da imagem, como as classes 0, 2, 4, 5, as quais são vestimentas superiores. As classe das calças 1 e vestidos 3 se parecem quanto a espessura, as classes de calçados 6, 7 e 9 se assemelham bastante em formato e tamanho, e a classe das bolsa 8 em alguns casos se assemelha às classes das roupas, quando as alças se confundem com o colarinho das camisas. Neste cenário, a rede enfrenta grande dificuldade em encontrar similaridades entre os elementos de uma mesma classe e construir os agrupamentos, ainda assim, o modelo consegue atingir resultados de acerto acima dos 5% em todos os grupo e em algums chega a valores superiores à 7%.

A partir da matriz de resultado do ST-DAC + 1 camada STN, já podemos observar a ação das camadas STN em vários agrupamentos, o maior ganho é na classe 5 de sandálias

Figura 14 – Matriz de confusão calculada a partir da taxa de acerto de *clustering*, dos modelos ST-DAC, na base Fashion MNIST




Fonte: o autor

e tamancos, as STNs encontraram transformações que aumentaram consideravelmente a taxa de acerto nesta classe de imagens, as sandálias que antes se confundiam majoritariamente com a classe de camisas 0, por existir pequenas camisas e com a classe de bolsas 8 pela alça e formato reduzido. Observamos em alguns casos que a rede rotaciona os objetos de classes distintas com estruturas semelhantes, para aumentar diferença extraclasse, o que pode ter ocorrido pelo fato das STNs em questão serem treinadas em um modelo de *clustering*, também otimizada para maximizar a variância extraclasse e aumentar a semelhança intraclasse. Desta forma, as sandálias podem ter aderido a uma rotação que se assemelhe a inclinação de um tamanco, aumentando a semelhança entre os elementos, e aumentando a divergência entre a rotação da camisa e da bolsa. A classe 1 da calça também se beneficiou bastante com as transformações, reduzindo a quantidade de falsos positivos com a classe 9, a qual representa botas. Houve também redução no desempenho de algumas classes, devido às transformações realizadas, mas os benefícios da STN foi percebido na maioria das classes, que apresentaram valores acima dos 6% atingindo até 8%.

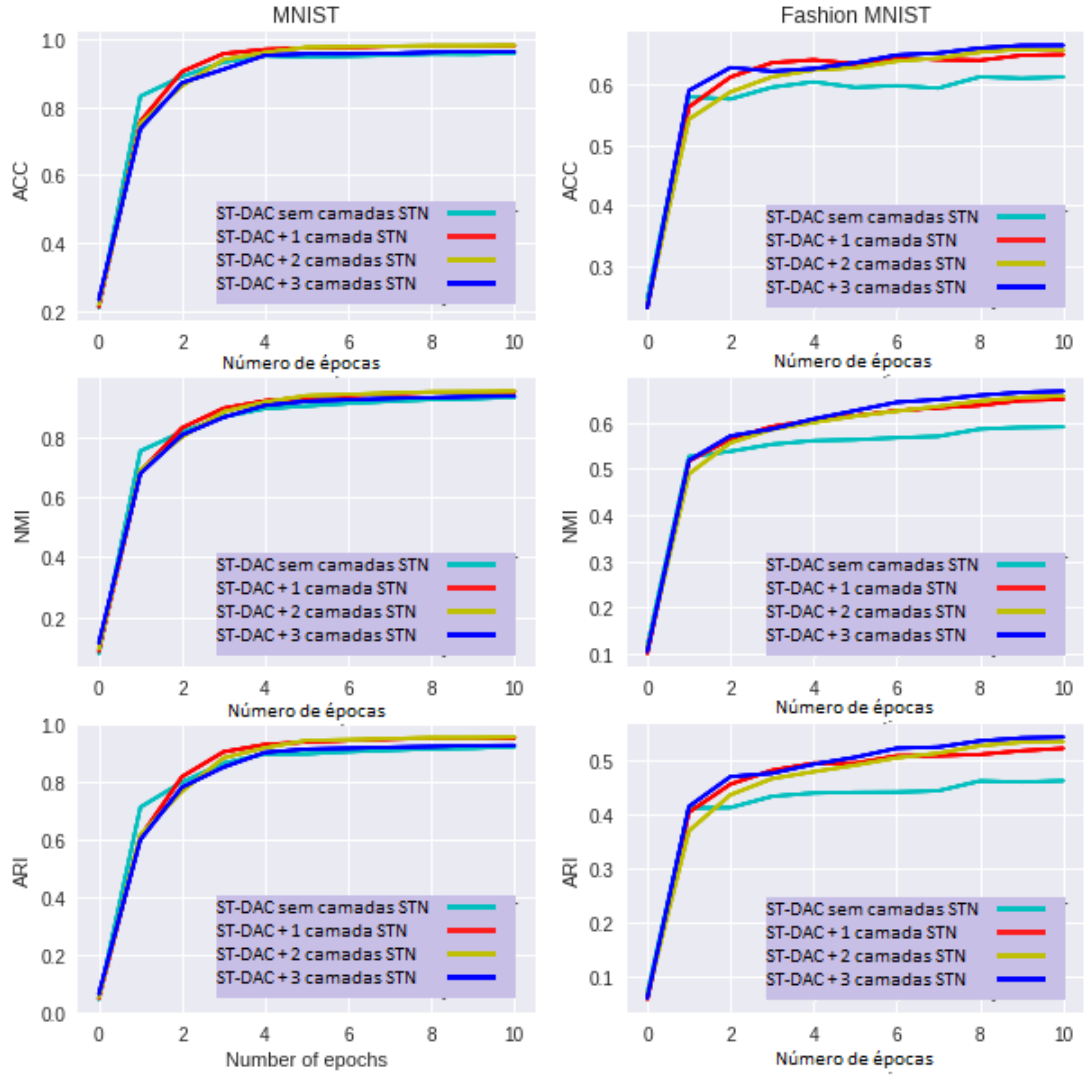
Verificando as matrizes do ST-DAC + 2 camadas STN e ST-DAC + 3 camadas STN, é possível notar que as taxas de acerto vão sendo equilibradas entre as classes, não ocasionando quedas, nem picos em classes específicas, mas aumentando os valores de acerto entre quase todas as casses em comparação ao modelo STN. As taxas de acerto ficam então em torno dos 6,5% à 7,5% na maioria das classes. Aumentando consideravelmente a taxa de acerto em relação ao modelo que não utiliza o mecanismo de atenção STN.

No processo de desenvolvimento do ST-DAC, encontramos algumas dificuldades em encontrar parâmetros e configurações que fizessem os modelos bases DAC* e STN trabalharem em sintonia, como mostrado no capitulo anterior. Logo, achamos interessante verificar a interação entre os mecanismos durante o treinamento. A Figura 15 apresenta a comparação das curvas de desempenho dos modelos com e sem camadas STN obtidas durante o treinamento. Através destas curvas percebemos que as STNs se beneficiaram bastante com o método adaptativo de seleção de amostras. Desde o primeiro momento de execução, é notável que as camadas STN trabalham em sintonia com a rede convolucional, não ocasionando perdas de desempenho, nem mesmo nas primeiras etapas de treinamento onde as *localization networks* nem a rede convolucional não estão suficientemente treinadas.

Após algumas primeiras épocas, os resultados são melhores que o modelo sem camadas STN. É possível perceber que, mesmo antes de convergir, as redes também se beneficiam das representações intermediárias obtidas pelas camadas STN durante o treinamento. As curvas permaneceram estáveis no conjunto de dados MNIST seguindo o mesmo padrão de crescimento sem declínios significativos no desempenho ao longo do tempo.

Analisando os resultados da base de dados do Fashion MNIST, vemos que as redes com uma e três camadas STN perdem desempenho em alguns pontos entre as épocas dois

Figura 15 – Comparação da performance de *clustering* entre modelos com diferentes números de camadas STN durante as épocas de treinamento no MNIST (esquerda) e Fashion MNIST (direita).




Fonte: o autor

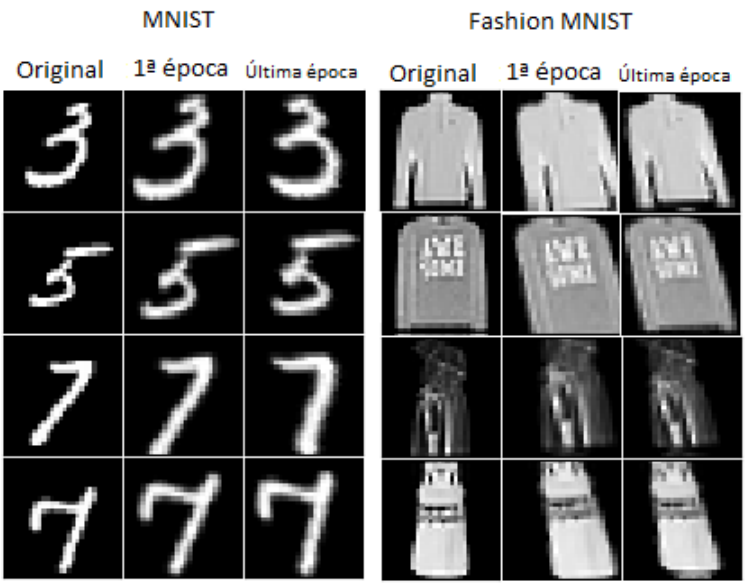
e seis e depois estabilizam e continuam a aumentar. Entre as combinações de camadas, o crescimento da rede com duas camadas STN permaneceu mais estável em comparação às demais durante todas as épocas de treinamento, apesar de ser superado no resultado final pela rede com 3 camadas STN.

Realizamos outros tipos de análise qualitativas, para avaliar se as alterações realizadas pelas STNs ajudaram na formação dos agrupamentos de dados. Além de colher informações a partir das imagens que fortaleçam as analises quantitativas mostradas anteriormente.

Primeiramente para analisar se as STNs estão trabalhando corretamente com as configurações propostas, extraímos algumas imagens da saída da primeira camada de transformação durante diferentes épocas de treinamento. Podemos observar as imagens na Figura 16.

A partir da Figura 16, observamos que as STNs agora apresentam o comportamento

Figura 16 – Comparação entre algumas imagens originais e suas respectivas saídas da primeira camada STN. Na coluna da esquerda é apresentada a imagem original, no centro a saída da primeira camada STN após a primeira época de treinamento e na coluna da direita a saída desta mesma após o treinamento.

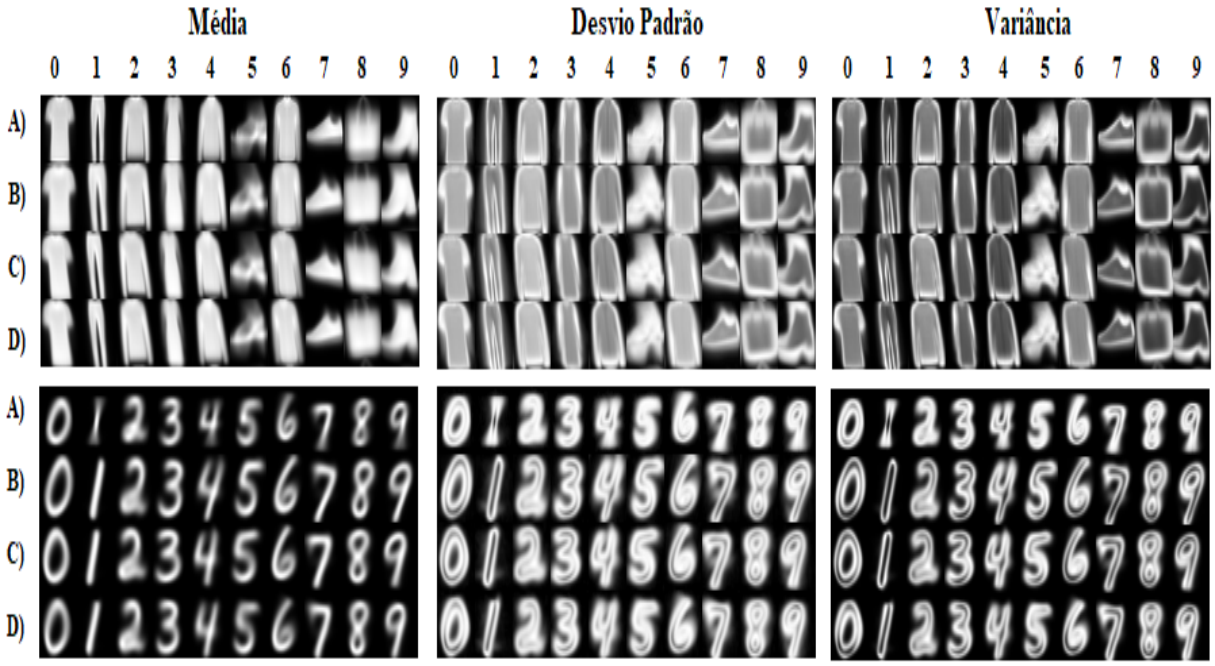



Fonte: o autor

esperado de acordo com o apresentado na maior parte dos trabalhos da literatura. A rede inicialmente aplica um zoom na imagem e ao longo do tempo reduz esse zoom para encaixar o objeto em uma região onde possa enquadrar melhor todos os detalhes da área de interesse, normalizando os objetos, corrigindo distorções e rotação durante o treinamento da rede, objetivando um melhor resultado para o modelo. No aprendizado do modelo, utilizamos *data augmentation* e as camadas aprenderam a corrigir várias transformações de rotação, escala e translação. Com este conhecimento as camadas STN giram os objetos para um ângulo padrão, no qual é possível ampliar e preencher uma área maior da imagem inteira sem perder grandes detalhes do objeto. As imagens de resultado de saída das camadas STN também apresentam um aspecto de desfoque e perdem alguns detalhes, mas essa perda é compensada pelas correções de transformação definidas anteriormente.

Na Figura 17, podemos observar as médias, desvios-padrões e variâncias entre as imagens de uma mesma classe, onde a linha a) corresponde as estatísticas processadas nas imagens originais das bases de dados e as demais linhas extraídas dos modelos ST-DAC com b) uma camada STN, c) duas camadas STN e d) 3 camadas STN. Todas as imagens foram extraídas após a conclusão do treinamento. Conseguimos observar, a partir das médias extraídas da base MNIST, que as transformações das STNs no modelo ST-DAC, deslocaram a imagem para uma região em comum o que é comprovado pelo nível de densidade dos dígitos nas imagens. Esta ação de normalização permite uma melhor comparação entre os elementos a partir das *features* extraídas. Além disso, as STNs, como visto em imagens anteriores, aplicaram uma forte aproximação dos dígitos o que pode

Figura 17 – Média, Desvio padrão e Variância entre todas as imagens de uma mesma classe da base Fashion MNIST na linha superior e da MNIST na inferior. As linhas apresentam as imagens extraídas da: A) base original, B) primeira camada STN do ST-DAC + 1 STN, C) primeira camada STN do ST-DAC + 2 STNs, D) primeira camada STN do ST-DAC + 3 STNs. O números acima das imagens representam os rótulos das classes. As imagens apresentadas do ST-DAC foram extraídas após a conclusão do treinamento.




Fonte: o autor

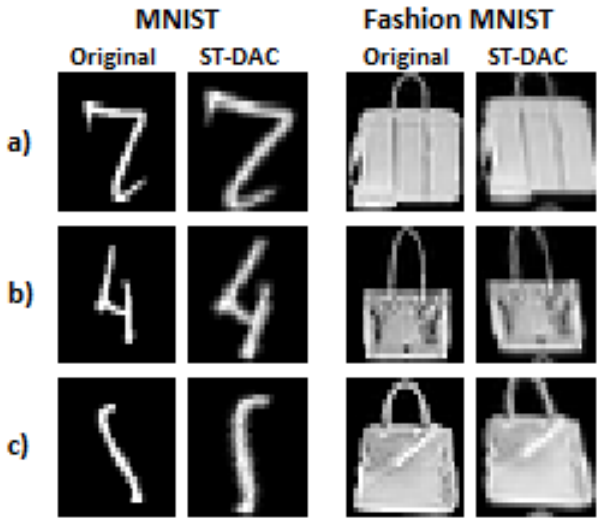
ter melhorado a captura de detalhes e o consequentemente os resultados. Analisando o desvio-padrão e a variância podemos notar um cenário como o anterior, na análise destas imagens levamos em consideração a espessura das bordas dos dígitos que mensuram a variância e o desvio-padrão. Observamos na grande maioria dos casos que no uso das STNs essas espessuras diminuíram o que indica uma maior normalização entre as posições e transformações dos objetos do grupo. Além disso, as lacunas internas onde havia uma densidade alta de variância e desvio-padrão foi diminuída, o que fortalece nossa afirmação.

Nesta mesma figura, agora realizando uma análise sobre a base de dados Fashion MNIST, a imagem das médias das classes aumentaram a densidade da mesma forma que na base MNIST, entretanto, de uma forma mais sútil o que é um bom sinal pois demostra o mesmo cenário anterior de alinhamento entre os objetos das classes. Além disso, as imagens sofreram transformações de rotação e zoom para que mais detalhes dos objetos fossem extraídos e representados nas *features*. Nas imagens de desvio padrão e variância conseguimos perceber uma redução de espessura das bordas, também de forma sútil, principalmente em algumas classes, em específico na 7, 8 e 9, o que demostra uma maior normalização entre os elementos, assim como um possível maior nível de homogeneidade de *clustering* entre os elementos destas classes.

Outro aspecto interessante é que durante o processo de treinamento do modelo ST-

DAC, as STNs realizam diversas transformações até encontrar um padrão que se enquadre melhor em normalizar as classes para uma determinada região. Entretanto, durante o processo a rede vai aprendendo com todas essas variações de transformações, aumentando sua capacidade de generalização, como se realizasse um *data augmentation* sobre os dados de forma controlada. No Apêndice A desta dissertação, encontram-se as imagens de média, desvio padrão e variância extraída em cada época de treinamento do modelo ST-DAC, a partir delas podemos observar este cenário.

Figura 18 – Imagens na coluna original foram agrupadas incorretamente, com a utilização do ST-DAC sem camadas STN. Na coluna ST-DAC são exibidas imagens agrupadas corretamente por modelo ST-DAC com camadas STN após a ação das mesmas.




Fonte: o autor

Para uma checagem mais detalhada coletamos alguns exemplos que evidenciam a ação das STNs em corrigir as imagens para que se enquadrem na classe correta. Na Figura 18 podemos ver alguns exemplos que foram classificados incorretamente pelo modelo ST-DAC sem camadas STN e nos modelos com o mecanismo de atenção foram corrigidos e atribuídos ao *cluster* correto. Em a) o digito 7 foi confundido originalmente com um 5, porém, após o zoom e a rotação aplicada, foi agrupado corretamente no *cluster* com elementos majoritariamente da classe 7. De forma semelhante, em b) o digito 4 foi classificado como 1, após o zoom e a rotação as diferenças entre as classes foi percebida pela rede e o exemplo foi agrupado corretamente. Em c) o digito 1 foi agrupado originalmente como 5 devido a inclinação, após a correção a semelhança com outros dígitos de sua mesma classe aumentou, devido ao alinhamento vertical. Na base Fashion MNIST extraímos exemplos da classe 8, os efeitos de correção são mais sutis, bolsas em a), b) e c) com características bem diversas são classificadas corretamente após o zoom da imagem, removendo um pouco a alça em alguns exemplos e aplicando um zoom, para alcançar um tamanho semelhante entre os elementos da mesma classe.

Finalmente, a partir de todas essas evidências, percebemos que ao tornar as redes

convolucionais robustas à transformações estruturais das amostras de imagens, as técnicas de atenção visual permitem que modelos com redes convolucionais mais simples obtenham resultados superiores a outros métodos de *deep image clustering*, como os obtidos por nosso modelo proposto ST-DAC.

# 5 CONCLUSÃO

Neste trabalho propomos uma novo modelo chamado *Spatial Transformer - Deep Adaptive Clustering* (ST-DAC) a partir da solução *Deep Adaptive Clustering*. Para isso, substituímos a rede de extração de características convolucionais originalmente presente no DAC* (CHANG et al., 2017) por um novo modelo mais simples combinado com camadas de *Spatial Transformer Networks*. Além disso, encontramos um conjunto de configurações, parâmetros e arquiteturas que compõem as redes STN que tornam a utilização do modelo mais eficiente e rápida superando o método DAC original (CHANG et al., 2017).

Avaliamos nossa abordagem ao realizar experimentos em dois bancos de dados públicos a base de dígitos MNIST e Fahion MNIST, e comparamos com outros métodos que se sobressaem na literatura no *image clustering*. O método ST-DAC foi comparado com DCN (YANG et al., 2016), VADE (JIANG et al., 2017), JULE (YANG; PARIKH; BATRA, 2016), DEC (XIE; GIRSHICK; FARHADI, 2016), DEC-DA (GUO et al., 2018), DAC (CHANG et al., 2017), DAC* (CHANG et al., 2017) e atingiu excelentes resultados. Com resultados superiores aos métodos da literatura na base de dados Fahion MNIST com uma boa margem de diferença e mesmo com uma rede convolucional mais simples do que a proposta originalmente no DAC se aproximou bastante do estado da arte no benchmark MNIST.

Também realizamos experimentos variando a quantidade de camadas STN no modelo convolucional proposto, para avaliar se, com a adição de novas camadas STN, usando a correção da transformação espacial sobre as características internas extraídas, os resultados do modelo podem melhorar. Os experimentos mostraram que nossa abordagem foi capaz de superar os outros métodos nas duas bases de dados avaliadas, alcançando promissores resultados em ambos os conjuntos de dados.

Além disso, para contribuir com a comunidade, disponibilizamos todos os códigos utilizados em nossos experimentos em um repositório público[1], com o intuito de fomentar a distribuição de conhecimento sobre o modelo proposto e para que novas pesquisas possam ser realizadas utilizando ideias ou componentes de nossa abordagem.

Os experimentos e a escrita desta dissertação contribuíram igualmente para a concepção e publicação de um artigo que resume em detalhes a concepção do modelo proposto ST-DAC assim como revisa outras propostas da área. O artigo se chamas *Improving Deep Image Clustering With Spatial Transformer Layers* (Souza Thiago 2017) e encontrasse disponível em "link arxivix"[2]. O artigo também foi submetido para o *International Joint Conference on Neural Networks* 2019.

Por fim, comprovamos que, com o uso de técnicas de atenção visual, como as camadas STN, o métodos de *deep image clustering* podem obter melhoria de desempenho a um

[1] https://github.com/tvmsouza/ST-DAC

[2] *Local reservado ao link do artigo, após aceitação de armazenamento no Arxiv

baixo custo e a partir desta estratégia o modelo ST-DAC conseguiu resultados promissores.

## 5.1 TRABALHOS FUTUROS

O uso de camadas STN apresentou resultados promissores na melhoria do desempenho do modelo ST-DAC. Como a área de atenção visual continua avançando e várias novas abordagens foram propostas, as quais ampliam a capacidade de camadas STN padrão, é natural que a sugestão para futuras análises foque em usar esses novos métodos para melhorar os modelos de *deep image clustering*.

Um destes métodos promissores é o apresentado em (DETLEFSEN; FREIFELD; HAUBERG, 2018), que define uma camada capaz de aprender e corrigir transformações diffeomórficas e combiná-las com as transformações afim ou homomórficas das camadas STN convencionais. Estas camadas são menos sensíveis a taxa de aprendizado e conseguem recuperar erros de transformação realizados nos dados da imagem. O trabalho (DETLEFSEN; FREIFELD; HAUBERG, 2018) já supera as camadas convencionais de STNs em alguns bancos de dados públicos na tarefa de classificação de imagens e também pode trazer melhorias nos resultados de *deep image clustering*.

Outro trabalho citado anteriormente que pode ser avaliado no modelo é o apresentado em (LIN; LUCEY, 2016), estas camadas propõem melhorar o alinhamento geométrico das transformações realizadas.

A combinação destas camadas citadas, inseridas no modelo ST-DAC pode trazer benefícios na qualidade das transformações aplicadas nos dados de entrada e *features* extraídas pelas camadas internas e nos parece uma abordagem promissora a melhorar os resultados de *image clustering*.

Para fazer com que as redes convolucionais trabalhassem em sintonia com STNs no modelo ST-DAC escolhemos uma arquitetura simples e com poucas camadas, para evitar o problema de *vanishing gradient*, entretanto seria interessante utilizar uma rede mais robusta e com mais camadas para atingir melhores resultados de reconhecimento, uma estratégia promissora seria utilizar uma *Residual Network* (ResNet) (HE et al., 2015) como rede convolucional, ou utilizar alguns de seus componentes. Essas redes já se mostraram bastante eficientes em constituir arquiteturas muito profundas, sem perda de informação entre as camadas, atingindo excelentes resultados em classificação de imagens.

Essas são algumas sugestões de trabalhos futuros. Entretanto a área de *Deep Image Clustering* e atenção visual estão em plena expansão, novas propostas e abordagens são publicadas frequentemente e com isso novas possibilidades de melhorias no modelo ST-DAC podem surgir de acordo com as novas pesquisas nas áreas em comum.

# REFERÊNCIAS

ABLAVATSKI, A.; LU, S.; CAI, J. Enriched deep recurrent visual attention model for multiple object recognition. 2017. Disponível em: <http://arxiv.org/abs/1706.03581>.

ALJALBOUT, E.; GOLKOV, V.; SIDDIQUI, Y.; CREMERS, D. Clustering with deep learning: Taxonomy and new methods. *CoRR*, abs/1801.07648, 2018. Disponível em: <http://arxiv.org/abs/1801.07648>.

ANTONOPOULOS, P.; NIKOLAIDIS, N.; PITAS, I. Hierarchical face clustering using sift image features. In: *2007 IEEE Symposium on Computational Intelligence in Image and Signal Processing*. [S.l.: s.n.], 2007. p. 325–329.

BA, J.; MNIH, V.; KAVUKCUOGLU, K. Multiple object recognition with visual attention. 2014. Disponível em: <http://arxiv.org/abs/1412.7755>.

BAY, H.; ESS, A.; TUYTELAARS, T.; GOOL, L. V. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, v. 110, n. 3, p. 346–359, jun. 2008.

BEDNAREK, M.; WALAS, K. Spatial transformations in deep neural networks. In: *2018 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*. [S.l.: s.n.], 2018. p. 158–162. ISSN 2326-0319.

BENGIO, Y.; LAMBLIN, P.; POPOVICI, D.; LAROCHELLE, H. Greedy layer-wise training of deep networks. In: *Proceedings of the 19th International Conference on Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 2006. (NIPS'06), p. 153–160. Disponível em: <http://dl.acm.org/citation.cfm?id=2976456.2976476>.

BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387310738.

CHANG, J.; WANG, L.; MENG, G.; XIANG, S.; PAN, C. Deep adaptive image clustering. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2017. p. 5880–5888.

CHENG, M.; MITRA, N. J.; HUANG, X.; TORR, P. H. S.; HU, S. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 37, n. 3, p. 569–582, March 2015. ISSN 0162-8828.

CHENG, Y.; ZHANG, T.; CHEN, S. Fast person-specific image retrieval using a simple and efficient clustering method. In: *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. [S.l.: s.n.], 2009. p. 1973–1977.

CîRSTEA, B.; LIKFORMAN-SULEM, L. Tied spatial transformer networks for digit recognition. In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. [S.l.: s.n.], 2016. p. 524–529. ISSN 2167-6445.

DETLEFSEN, N. S.; FREIFELD, O.; HAUBERG, S. Deep diffeomorphic transformer networks. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2018. p. 4403–4412.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996. (KDD'96), p. 226–231. Disponível em: <http://dl.acm.org/citation.cfm?id=3001460.3001507>.

FRINTROP, S. Computational visual attention. *Computer Analysis of Human Behavior*, Springer, p. 69–101, 2011.

GOODFELLOW, I.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; BENGIO, Y. Generative adversarial nets. In: GHAHRAMANI, Z.; WELLING, M.; CORTES, C.; LAWRENCE, N. D.; WEINBERGER, K. Q. (Ed.). *Advances in Neural Information Processing Systems 27*. [S.l.: s.n.], 2014. p. 2672–2680.

GUO, X.; ZHU, E.; LIU, X.; YIN, J. Deep embedded clustering with data augmentation. In: *The 10th Asian Conference on Machine Learning (ACML)*. [S.l.: s.n.], 2018.

HALOI, M. Traffic sign classification using deep inception based convolutional networks. *CoRR*, abs/1511.02992, 2015.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. Disponível em: <http://arxiv.org/abs/1512.03385>.

JADERBERG, M.; SIMONYAN, K.; ZISSERMAN, A.; KAVUKCUOGLU, k. Spatial transformer networks. In: CORTES, C.; LAWRENCE, N. D.; LEE, D. D.; SUGIYAMA, M.; GARNETT, R. (Ed.). *Advances in Neural Information Processing Systems 28*. [S.l.: s.n.], 2015. p. 2017–2025.

JIANG, Z.; ZHENG, Y.; TAN, H.; TANG, B.; ZHOU, H. Variational deep embedding: An unsupervised and generative approach to clustering. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. [S.l.: s.n.], 2017. (IJCAI'17), p. 1965–1972.

KARYPIS, G.; HAN, E.-H.; KUMAR, V. Chameleon: hierarchical clustering using dynamic modeling. *Computer*, v. 32, n. 8, p. 68–75, Aug 1999. ISSN 0018-9162.

KIMURA, A. *Presentation slides for International Symposium on Brainware LSI in Tohoku University: Computational models of human visual attention driven by auditory cues*. 2015. <https://www.slideshare.net/akisatokimura/computational-models-of-human-visual-attention-driven-by-auditory-cues>.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. Disponível em: <http://arxiv.org/abs/1412.6980>.

KINGMA, D. P.; WELLING, M. Auto-encoding variational bayes. 2013.

KUHN, H. W. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, v. 52, n. 1, p. 7–21, 2005.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, Nov 1998.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, Nov 1998. ISSN 0018-9219.

LEE, Y.; AHN, H.; RHEE, S. Efficient image retrieval using advanced clustering surf. In: *2012 15th International Conference on Network-Based Information Systems*. [S.l.: s.n.], 2012. p. 749–753.

LIN, C.; LUCEY, S. Inverse compositional spatial transformer networks. *CoRR*, abs/1612.03897, 2016. Disponível em: <http://arxiv.org/abs/1612.03897>.

LLOYD, S. P. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, v. 28, p. 129–137, 1982.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, v. 60, n. 2, p. 91–110, nov. 2004.

MARQUES, J. S. *Reconhecimento de Padrões Métodos Estatísticos e Neuronais*. 2. ed. [S.l.]: IST Press, 2005.

MARQUES, J. S. *Computação Gráfica*. 11. ed. The address: Elsevier, 2008. v. 2.

MEI, L.; GUO, X.; YIN, W. Learning geometric invariance features and discrimination representation for image classification via spatial transform network and xgboost modeling. In: *Proceedings of the 7th International Conference on Informatics, Environment, Energy and Applications*. New York, NY, USA: ACM, 2018. (IEEA '18), p. 222–226. ISBN 978-1-4503-6362-4. Disponível em: <http://doi.acm.org/10.1145/3208854.3208886>.

MIN, E.; GUO, X.; LIU, Q.; ZHANG, G.; CUI, J.; LONG, J. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, v. 6, p. 39501–39514, 2018.

MIRONICă, I.; IONESCU, B.; VERTAN, C. Hierarchical clustering relevance feedback for content-based image retrieval. In: *2012 10th International Workshop on Content-Based Multimedia Indexing (CBMI)*. [S.l.: s.n.], 2012. p. 1–6.

MNIH, V.; HEESS, N.; GRAVES, A.; KAVUKCUOGLU, K. Recurrent models of visual attention. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. [S.l.: s.n.], 2014. (NIPS'14), p. 2204–2212.

NAGAVI, T. C.; ANUSHA, S. B.; MONISHA, P.; POORNIMA, S. P. Content based audio retrieval with mfcc feature extraction, clustering and sort-merge techniques. In: *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*. [S.l.: s.n.], 2013. p. 1–6.

QI, G.; HUA, X.; RUI, Y.; TANG, J.; MEI, T.; ZHANG, H. Correlative multi-label video annotation. In: *Proceedings of the 15th ACM International Conference on Multimedia*. New York, USA: [s.n.], 2007. p. 17–26.

RUBLEE, E.; RABAUD, V.; KONOLIGE, K.; BRADSKI, G. Orb: An efficient alternative to sift or surf. In: *Proceedings of the 2011 International Conference on Computer Vision*. [S.l.: s.n.], 2011. (ICCV '11), p. 2564–2571.

SARAVANAN, D. Cure clustering technique suitable for video data retrieval. In: *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. [S.l.: s.n.], 2016. p. 1–4.

SCLAROFF, S.; CASCIA, M.; SETHI, S.; TAYCHER, L. Unifying textual and visual cues for content-based image retrieval on the world wide web. *Comput. Vis. Image Underst.*, v. 75, p. 86–98, jul. 1999.

SILESHI, M.; GAMBACK, B. Evaluating clustering algorithms: Cluster quality and feature selection in content-based image clustering. In: *2009 WRI World Congress on Computer Science and Information Engineering*. [S.l.: s.n.], 2009. v. 6, p. 435–441.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. 2014. Disponível em: <https://arxiv.org/pdf/1409.1556.pdf>.

SPRINGENBERG, J. T. Unsupervised and semi-supervised learning with categorical generative adversarial networks. 2015. Disponível em: <https://arxiv.org/pdf/1511.06390.pdf>.

SPRINGENBERG, J. T.; DOSOVITSKIY, A.; BROX, T.; RIEDMILLER, M. Striving for simplicity: The all convolutional net. 2014. Disponível em: <https://arxiv.org/pdf/1412.6806.pdf>.

SUN, Y.; FISHER, R. Object-based visual attention for computer vision. *Artif. Intell.*, Elsevier Science Publishers Ltd., Essex, UK, v. 146, n. 1, p. 77–123, maio 2003. ISSN 0004-3702. Disponível em: <http://dx.doi.org/10.1016/S0004-3702(02)00399-5>.

TIELEMAN, T.; HINTON, G. *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*. 2012. COURSERA: Neural Networks for Machine Learning.

VINCENT, P.; LAROCHELLE, H.; LAJOIE, I.; BENGIO, Y.; MANZAGOL, P.-A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, JMLR.org, v. 11, p. 3371–3408, dez. 2010. ISSN 1532-4435. Disponível em: <http://dl.acm.org/citation.cfm?id=1756006.1953039>.

VU, H. T.; HUANG, C. A multi-task convolutional neural network with spatial transform for parking space detection. In: *2017 IEEE International Conference on Image Processing (ICIP)*. [S.l.: s.n.], 2017. p. 1762–1766. ISSN 2381-8549.

WANG, F.; TAX, D. M. J. Survey on the attention based RNN model and its applications in computer vision. *CoRR*, abs/1601.06823, 2016. Disponível em: <http://arxiv.org/abs/1601.06823>.

WANG, J.; WANG, J.; KE, Q.; ZENG, G.; LI, S. Fast approximate k-means via cluster closures. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2012. p. 3037–3044.

WANG, J.; WANG, J.; SONG, J.; XU, X.; SHEN, H. T.; LI, S. Optimized cartesian k-means. *IEEE Transactions on Knowledge and Data Engineering*, v. 27, n. 1, p. 180–192, Jan 2015.

WAZARKAR, S.; KESHAVAMURTHY, B. N. A survey on image data analysis through clustering techniques for real world applications. *Journal of Visual Communication and Image Representation*, v. 55, p. 596 – 626, 2018. ISSN 1047-3203. Disponível em: <http://www.sciencedirect.com/science/article/pii/S104732031830172X>.

XIAO, H.; RASUL, K.; VOLLGRAF, R. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017.

XIE, J.; GIRSHICK, R.; FARHADI, A. Unsupervised deep embedding for clustering analysis. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. [S.l.: s.n.], 2016. (ICML'16), p. 478–487.

XU, D.; TIAN, Y. A comprehensive survey of clustering algorithms. *Annals of Data Science*, v. 2, n. 2, p. 165–193, Jun 2015. ISSN 2198-5812. Disponível em: <https://doi.org/10.1007/s40745-015-0040-1>.

YANG, B.; FU, X.; SIDIROPOULOS, N. D.; HONG, M. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. 2016. Disponível em: <http://arxiv.org/abs/1610.04794>.

YANG, J.; PARIKH, D.; BATRA, D. Joint unsupervised learning of deep representations and image clusters. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. p. 5147–5156.

ZHANG, D.; RAO, Y.; ZHAO, J.; ZHAO, J.; HU, A.; CAI, B. Feature based segmentation and clustering on forest fire video. In: *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. [S.l.: s.n.], 2007. p. 1788–1792.

ZHANG, W.; WU, X.; ZHU, W.; YU, L. Unsupervized image clustering with sift-based soft-matching affinity propagation. *IEEE Signal Processing Letters*, v. 24, n. 4, p. 461–464, April 2017.

ZHANG, Y.; MIAO, Z. Object recognition based on orb and self-adaptive kernel clustering algorithm. In: *2014 12th International Conference on Signal Processing (ICSP)*. [S.l.: s.n.], 2014. p. 1397–1402.

# APÊNDICE A – IMAGENS ESTATÍSTICA DE SAÍDA DAS CAMADAS STN DURANTE O PERÍODO DE TREINAMENTO DO ST-DAC

Figura 19 – Média, desvio Padrão e variância das classes de imagens da base MNIST, extraídas da primeira camada STN de cada modelo ST-DAC. Durante todas as épocas de treino.

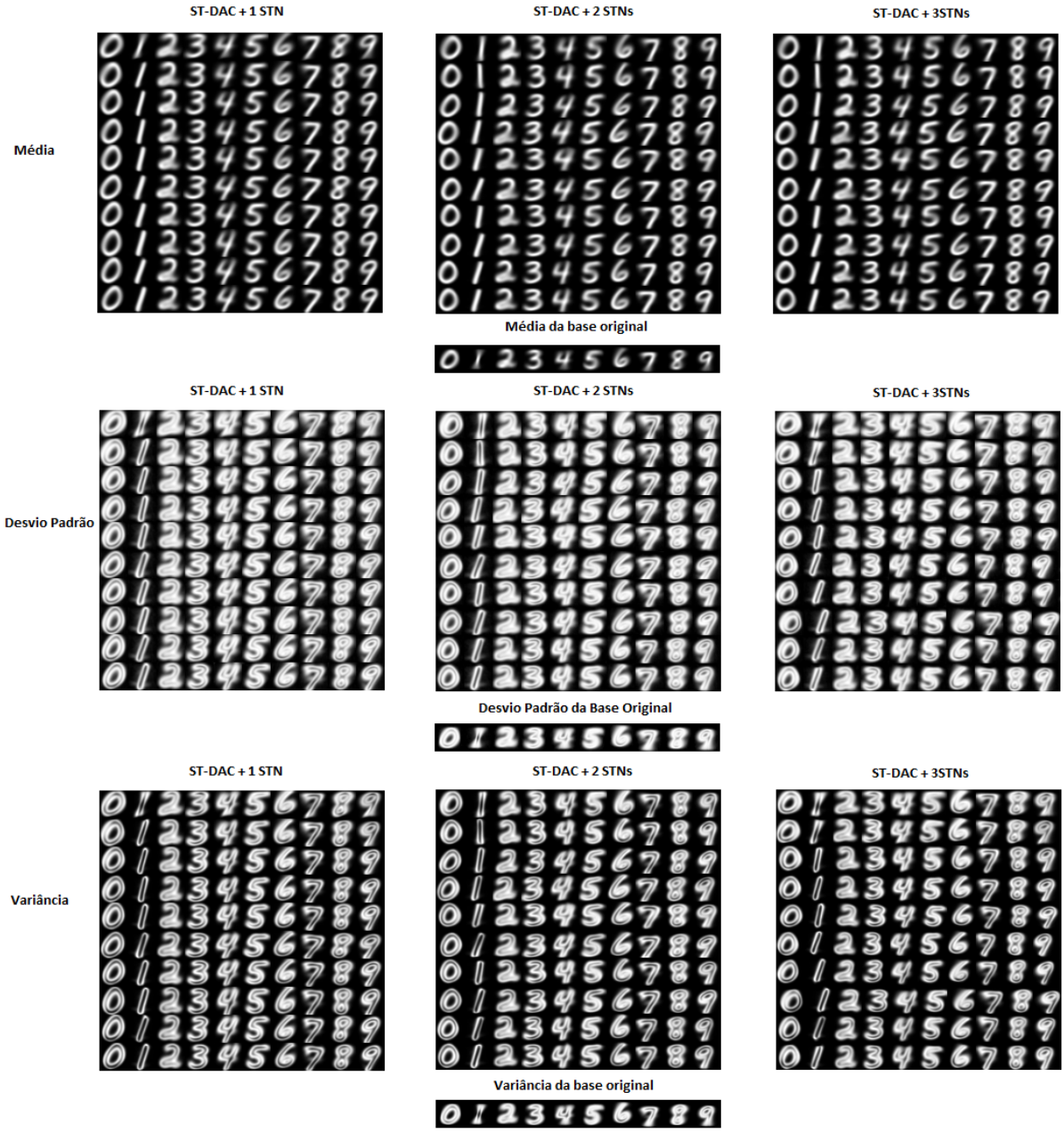


Figura 20 – Média, desvio Padrão e variância das classes de imagens extraídas da base Fashion MNIST da primeira camada STN de cada modelo ST-DAC. Durante todas as épocas de treino.