Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Engenharia Elétrica

USO DE AUTOCODIFICADORES CONVOLUTIVOS NA CLASSIFICAÇÃO DE REPRESENTAÇÕES ICÔNICAS PARA OBJETOS RÍGIDOS

PROJETO DE PESQUISA II

CHRISTIANE RAULINO ALMEIDA MOLINA

ORIENTADORES:

RAIMUNDO CARLOS SILVÉRIO FREIRE

JUGURTA ROSA MONTALVÃO FILHO

Campina Grande - PB, dezembro de 2022

USO DE AUTOCODIFICADORES CONVOLUTIVOS NA CLASSIFICAÇÃO DE REPRESENTAÇÕES ICÔNICAS PARA OBJETOS RÍGIDOS

Christiane Raulino Almeida Molina

Relatório apresentado ao Programa de Pós-graduação em Engenharia Elétrica - PPgEE, da Universidade Federal de Capina Grande, como parte dos requisitos necessários de avaliação do componente Projeto e Pesquisa II.

Orientadores:
Raimundo Carlos Silvério Freire
Jugurta Rosa Montalvão Filho

Campina Grande - PB, dezembro de 2022

Sumário

1 INTRODUÇÃO

Plantas elétricas de alta potência necessitam constantemente de supervisão, a fim de garantir seu correto funcionamento e detectar possíveis falhas antes mesmo de seus equipamentos danificarem. Nesse contexto, métodos de monitoramento automatizados [1, 2] compõem uma solução com o mínimo de intervenção e recursos humanos no espaço físico operante, capazes de perceber não conformidades o mais rápido possível, evitando danos ao processo de produção e distribuição. No âmbito de visão computacional, imagens adquiridas de plantas elétricas de alta potência através de *drones* ou de captura manual remota geram dados que, ao serem analisados, podem trazer um preciso diagnóstico de problemas em curso, assim como a antecipação de problemas futuros, como é o caso da inspeção e manutenção de isoladores [3, 4], indicando a necessidade de manutenções preventivas ou substituição de equipamentos com vida útil limítrofe, por exemplo. Desta maneira, a utilização de técnicas de visão computacional pode fornecer alternativas aos sistemas de monitoramento baseados em sinais elétricos ou mesmo trabalhar em conjunto com eles na construção de um monitoramento mais robusto.

A aplicação de interesse que inspirou este projeto de pesquisa consiste em perceber não-conformidades de operação em equipamentos presentes na área de alta potência em uma termelétrica, onde, definida como tarefa inicial realizada em projeto prévio a este, um equipamento elétrico é isolado e classificado de acordo com suas características através da imagem capturada. Baseados nessa tarefa, experimentos anteriores evidenciaram que as imagens adquiridas são de difícil manipulação e análise, pois os equipamentos a serem classificados são fotografados em diferentes perspectivas e há bastante heterogeneidade no ambiente em que o objeto de interesse está presente. Todavia, apesar do alto grau de complexidade das imagens (i.e, fundo heterogêneos, outros equipamentos na planta, diferentes perspectivas, diferença de iluminação), pôde-se perceber que a estrutura principal do equipamento não se altera. Em outras palavras, a estrutura é rígida, pois não há deformação no formato principal do equipamento, o que não acontece em problemas clássicos de reconhecimento, como o de reconhecimento de dígitos manuscritos [5] em que parte do processamento é voltada à busca de características invariantes a essas deformações.

Motivado por essa característica dos equipamentos a serem classificados no problema de interesse, e baseado no alto desempenho apresentado pelas arquiteturas de redes neuronais convolutivas para processamento de imagens, este projeto de pesquisa propõe o uso de autocodificadores convolutivos a fim identificar as características relevantes que definem a estrutura rígida do equipamento e sua posição/orientação na imagem. Ao realizar o treinamento da rede com imagens sintéticas simplificadas (representações icônicas), era esperado que as características provenientes desse modelo

representassem as variáveis latentes do possível *manifold* do fenômeno associado ao tipo e posição/orientação do equipamento.

Dessa forma, foi realizada inicialmente uma revisão bibliográfica de métodos de análise de componentes e aprendizado de *manifolds*, baseados em autocodificadores, bem como a implementação de autocodificadores convolutivos aplicados a bases públicas como a MNIST, utilizada em [6].

Além disso, após o estudo teórico e experimentações em base pública, foi sintetizada uma base artificial de estruturas simplificadas (ícones), inspirada na rigidez de formato dos equipamentos presentes em termelétricas (isoladores, transformadores e chaves, por exemplo). As imagens foram construídas a partir do que se espera que sejam as variáveis latentes (posição e orientação do objeto). Desta maneira, sintetizado artificialmente a dimensionalidade do fenômeno, foi experimentada a possibilidade do gargalo do autocodificador apresentar as variáveis usadas para construir a imagem de entrada. Em outras palavras, o vetor de saída do codificador e o vetor de transformações, usados para sintetizar o dado de entrada, foram comparados, realizando um aprendizado explícito do mapeamento da rede neural convolutiva, que seria desconhecido no caso de uma imagem real como entrada.

Em seguida, com a base formada, foram realizados experimentos utilizando os dados artificiais baseados na aplicação de interesse, descritos anteriormente. Tal experimentação proporcionou a verificação da viabilidade do uso de autocodificadores convolutivos para aprendizado explícito de variáveis latentes.

1.0.1 Objetivo Geral

O objetivo geral dessa proposta é investigar as características da representação de dimensão reduzida geradas no gargalo de autocodificadores convolutivos, em que o treinamento da rede é realizado com imagens sintéticas inspiradas em objetos com estrutura principal rígida, mais especificamente de equipamentos elétricos presentes em usinas termelétricas.

1.0.2 Objetivos Específicos:

- Estudar a teoria da estrutura dos autocodificadores convolutivos, assim como sua implementação em base pública [5];
- Criar a base sintética, inspiradas nas estruturas rígidas dos equipamentos presentes nas imagens capturadas em uma usina termelétrica instalada no estado de Sergipe (o acesso a esses dados é garantido, através de projeto de colaboração já em andamento);

- Aplicar a rede implementada e analisar os resultados, a fim de observar a representação das variáveis latentes dos objetos simplificados criados;
- Produzir e publicar artigo científico, em periódico e anais de eventos, contendo os resultados obtidos neste projeto.

1.1 Estrutura do Projeto de Pesquisa

A sequência deste relatório de pesquisa está estruturada da seguinte maneira:

Capítulo 2 - é apresentada uma revisão bibliográfica no tocante ao uso de autocodificadores e redes neuronais convolutivas aplicadas à tarefas de visão computacional;

Capítulo 3 - uma fundamentação teórica é apresentada acerca de autocodificadores e redes neurais, assim como matrizes de transformação e algoritmo de desenho linha;

Capítulo 4 - descreve-se a abordagem proposta, através das etapas desenvolvidas ao longo do projeto, a fim de atingir os objetivos da disciplina;

Capítulo 5 - apresentam-se a descrição dos experimentos realizados, resultados e discussão dos mesmos;

Capítulo 6 - são discutidas as conclusões e expectativas para trabalhos futuros.

A seguir é apresentada uma revisão bibliográfica em que são abordadas características relevantes no campo de visão computacional quanto ao uso de autocodificadores, redes neurais e trabalhos relacionados.

2 REVISÃO BIBLIOGRÁFICA

Visão computacional é um dos grandes de campos de aprendizado de máquina e possui métodos capazes de criar modelos matemáticos para tomada de decisões, de acordo com sua aplicação. A tarefa de reconhecimento de objetos em uma imagem é uma de suas aplicações bastante conhecida e possui diversas abordagens, desde as mais antigas como máquina de vetores de suporte [7] até estudos mais recentes, como os baseados em redes neuronais convolutivas [8] e em mecanismos de atenção [9]. Muitas dessas abordagens utilizadas para reconhecimento são fundamentadas na hipótese de que imagens são consideradas pontos em um espaço de alta dimensão (onde cada pixel representa uma coordenada) e cada fenômeno a ser reconhecido, a partir de imagens, ocupa um subespaço de menor dimensão, nomeado espaço latente e definido pelo menor número de

características essenciais (independentes) possíveis que represente esse fenômeno (variáveis latentes). Portanto, mapear o espaço de alta dimensão em uma variedade (*manifold*, em inglês) [10] de dimensionalidade intrínseca reduzida é uma das maneiras de representar de forma significativa os dados de interesse e que pode ser usada de diversas maneiras, como, por exemplo, facilitar o sistema de reconhecimento de objetos.

Apesar de ser uma tarefa usual, encontrar uma estrutura de baixa dimensão incorporada (no inglês, *embedded*) no ambiente de alta dimensão no qual os dados se encontram é um desafio ainda bastante discutido e compõe a área de aprendizado de *manifold* [10]. Vale ressaltar que cada fenômeno associado a um conjunto de dados possui seu respectivo espaço latente e, a depender de que tipo de informação se deseja obter no conjunto, os *manifolds* almejados podem ser diferentes, até mesmo para uma mesmo base de dados. Caso a estrutura do *manifold* seja linear (dados incorporados em um hiperplano), técnicas de análise de componentes como Análise de Componentes Principais (em inglês *Principal component analysis* - PCA) [11] e como *Multidimensional scaling* (MDS) [12] são aplicáveis. Para o caso da natureza dos dados gerar complexas relações entre as variáveis latentes, formando uma estrutura não-linear, técnicas de aprendizado não-lineares são necessárias, como *Isometric Feature Mapping* (ISOMAP)[13], *locally linear embedding* (LLE) [14], autocodificadores [15], entre outras.

Considerando a complexidade das relação entre pixels em uma imagem, as características essenciais deste tipo de dado geram estruturas curvas (não lineares). Sendo assim, os autocodificadores podem ser considerados uma boa alternativa para aprendizado *manifolds* não-lineares em imagens.

Autocodificadores podem ser implementados na forma de rede neural utilizadas em aprendizado não supervisionado, possuindo duas etapas: a codificação, que representa o dado de maneira condensada; e a decodificação, que reconstrói o dado a partir da representação reduzida na saída da etapa anterior, o codificador. O treinamento é realizado ajustando os parâmetros da rede com o objetivo de se obter a entrada original na saída do decodificador. Dentre os diversos tipos e aplicações existentes de autocodificadores [16], pode-se destacar os convolutivos aplicados à tarefa de classificação de imagens [17][18][19]. Neste caso, a saída do codificador (também chamada de gargalo do codificador, por possuir uma dimensão reduzida) é usada como um extrator de características essenciais e, então, conectada a um classificador qualquer.

Redes neurais convolutivas (*Convolutional Neural Networks* - CNN ou ConvNets) vêm sendo desenvolvidas, nos últimos 10 anos, com bons resultados processamento de imagens. Implementadas com sucesso na década de 90 por Lecun [6], esse tipo de rede neural foi inspirada no sistema de visão biológico [20], onde a imagem é processada por sensibilidade local, ou seja, por regiões. Sendo assim, ao invés de tratar a imagem inteira para entrada da rede neural, como uma *Multi Layer Perceptron* (MLP) faria, as

CNNs processam a imagem através de filtros (também chamados de *kernels*) treinados para capturar diferentes características da imagem (e.g. bordas, formas geométricas e sombras). Há diversos tipos de ConvNets bem estabelecidas [21][22][23][24], e suas aplicações estão presentes em diversas áreas de visão computacional, como auxílio em diagnóstico por análise de imagens [25], e na detecção de equipamentos em um planta elétrica [26][27][28][29], para citar alguns exemplos. Os resultados apresentados nos últimos 10 anos firmaram as ConvNets no campo de visão computacional.

Com efeito, de acordo com as considerações científicas apresentadas, desenvolver aprendizado de *manifold*, através da extração de variáveis latentes do gargalo de um autocodificador convolutivo, caracteriza-se como um caminho promissor para o reconhecimento robusto de imagens e de grande potencial de pesquisa.

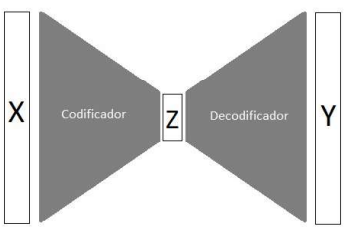
3 FUNDAMENTAÇÃO TEÓRICA

A fim de estabelecer uma fundamentação teórica para entendimento dos experimentos realizados durante o projeto de pesquisa, nesta seção serão apresentados os conceitos relevantes para esse trabalho acerca de métodos para detecção das representações icônicas como autocodificadores, das redes neurais perceptrons de múltipla camada (em inglês, *Multilayer perceptron* - MLP), das redes neuronais convolutivas, e de métodos de computação gráfica para criação da base de representações, como o algoritmo de desenho de linha e matrizes de transformações.

3.1 Autocodificadores

Um autocodificador é um método de aprendizado não supervisionado cuja estrutura possui duas partes principais: codificador e decodificador 1. O codificador tem como objetivo representar o dado de entrada X em uma dimensão reduzida Z, extraindo suas características principais. O decodificador reconstrói o padrão de entrada a partir da representação reduzida gerada na saída do codificador, produzindo a saída final Y da estrutura.

Figura 1 – *Autocodificador*



A saída do codificador, que é também a entrada do decodificador, forma uma espécie de gargalo de dimensões, na qual há uma redução significativa da dimensão dos dados de entrada. Desta maneira, um dado de dimensão m é mapeado pelo codificador em um espaço de dimensão n, onde $n \ll m$, através de uma função $G(X)$. O decodificador, por sua vez, mapeia através de $H(Z)$ a saída do codificador no espaço original de entrada do autocodificador.

A formação mais utilizada para esse método consiste na utilização de redes neurais determinísticas (e.g redes MLP e ConvNets), treinando-as de maneira não-supervisionada e fazendo com que a estrutura seja capaz de reconstruir em Y a entrada X, a menos de um erro de reconstrução que se deseja o menor possível.

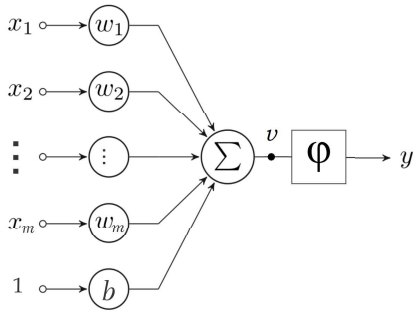
3.2 Rede MLP

Considerada um dos tipos de redes neurais mais clássico, a *Multilayer Perceptrons* é tipicamente formada por elementos básicos chamados perceptrons, e para um bom entendimento de seu funcionamento, primeiramente faz-se necessária a definição dessas unidades.

3.2.1 Perceptron

O perceptron é um modelo matemático simples, que realiza uma combinação entre a entrada e seus parâmetros (pesos sinápticos e viés), representada na figura 2.

Figura 2 – *Perceptron*



Em que a saída parcial v, a partir de uma entrada de dimensão m, pode ser escrita na forma:

$$v = \sum_{i=1}^{m} w_i x_i + b \tag{3.1}$$

Por uma perspectiva de classificação, esse neurônio matemático é capaz de definir um hiperplano ($\sum_{i=1}^{m} w_i x_i + b = 0$) como fronteira de decisão, separando o espaço m-dimensional (definido pelas variáveis da entrada) em em regiões.

Aplicada à saída parcial v, a função de ativação φ é utilizada para deformar não-linearmente v e gerar a a amplitude de saída y de um neurônio. Geralmente, as funções adotadas para restrição normalizam a saída em intervalos fechados entre 0 e 1, ou entre -1 e 1. A saída do perceptron pode ser escrita, então, pela equação 3.2.

$$y = \varphi(\sum_{i=1}^{m} w_i x_i + b) \tag{3.2}$$

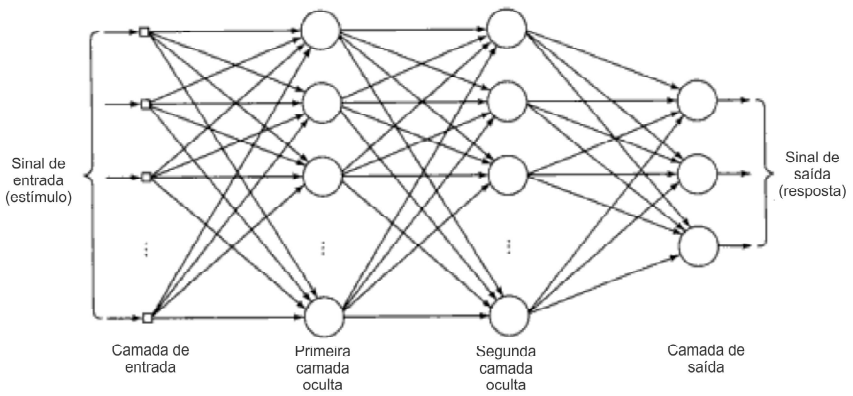
Há diversas funções de ativação capazes de normalizar a saída de um neurônio: sigmóide, tangente hiperbólica e unidade retificadora linear (ReLu), para citar alguns exemplos. Contudo, o melhor tipo de ativação utilizado no neurônio artificial depende do fenômeno a ser modelado. Em outras palavras, é escolhida a função de ativação que garante o melhor desempenho ao modelo. Por exemplo, é bastante comum utilizar funções ReLu como função de ativação para os perceptrons em imagens como dado de entrada.

Em resumo, o *perceptron* é a unidade básica de uma MLP e possui três parâmetros que o definem: vetor de pesos sinápticos(w_i, onde $i = 1, 2, ..., m$), o viés (b) e o tipo da função de ativação (φ).

3.2.2 Arquitetura da MLP

Uma vez definida a unidade básica, uma MLP é uma rede com múltiplas camadas formadas por essas unidades, representadas por círculos na figura 3. Geralmente, sua arquitetura é composta por uma camada de entrada com dados a serem processados (sinal de estímulo), por camadas intermediárias (ou ocultas) e por uma camada de saída (resposta), que gera a informação útil de acordo com a tarefa a ser desenvolvida.

Figura 3 – Grafo arquitetural de um perceptron de múltiplas camadas com duas camadas ocultas [30]



No sentido da propagação da informação, os dados a serem processados são o sinal de entrada da primeira camada intermediária, e sua saída serve com entrada da segunda camada intermediária, seguindo o mesmo fluxo até chegar o final da rede. Logo,

os neurônios de uma camada têm com entrada apenas os sinais de saída da camada anterior, estabelecendo um fluxo sequencial e caracterizando a rede de alimentação direta (em inglês *feed forward*). O fluxo de operações para a arquitetura exibida em 3 está representado pelo algoritmo 1 em pseudocódigo.

Algorithm 1 Caminho direto da MLP de duas camadas intermediárias

$X \leftarrow Entrada$
$X \leftarrow [X\ 1]$ ▷ Inclusão do víes
$A \leftarrow W_1.X$ ▷ W_1 Pesos sinápticos e vieses dos neurônios da primeira camada oculta
$D \leftarrow \varphi(A)$
$D \leftarrow [D\ 1]$ ▷ Inclusão do víes
$B \leftarrow W_2.D$ ▷ W_2 Pesos sinápticos e vieses dos neurônios da segunda camada oculta
$Y \leftarrow \varphi(B)$

Outro aspecto relevante observado na figura 3 é que cada neurônio de uma camada está conectado a todos os neurônios da camada anterior. Portanto, pode-se dizer que se trata de uma RNA totalmente conectada. Dessa maneira, se uma camada possui C_2 neurônios e está conectada a uma camada de entrada com C_1, cada neurônio de C_2 possui $C_1 + 1(viés)$ graus de liberdades pra definir sua fronteira de classificação. Considerando todos os neurônios da camada, há $C_2.C_1 + C_2$ parâmeros livres para ajuste das fronteiras.

Com a arquitetura definida, no projeto da rede MLP, para um dado m-dimensional e uma saída de K classes, define-se primeiramente o número de camadas intermediárias (I) e o número de neurônios de cada camada (N_{C_i}, onde $i = 1, 2, ..., I$). O número de camadas intermediárias e o número de neurônios de cada camada faz parte do projeto da rede, e são parâmetros a serem definidos para gerar um melhor desempenho na tarefa executada pela RNA. Em seguida os parâmetros livres de cada camada (pesos sinápticos e vieses dos neurônios) são ajustados, através de um algoritmo de aprendizado (treinamento), para cumprir o objetivo desejado (e.g, reconhecer um equipamento elétrico em uma imagem).

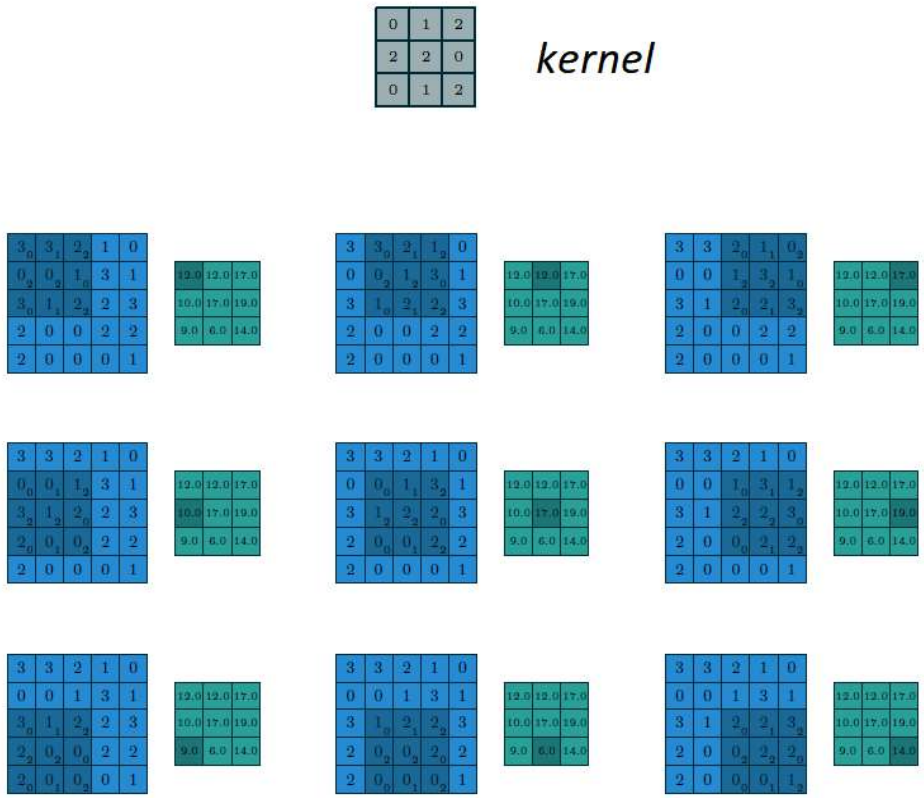
3.3 Rede Convolutiva

Redes convolutivas são um caso particular da MLP, e a principal particularidade desse tipo de rede está na substituição da camada totalmente conectada por uma camada de pesos compartilhados. A ideia de compartilhamento de pesos parte da motivação biológica em utilizar o conceito de campos receptivos locais [31], em que a camada convolutiva é capaz de reconhecer formas bidimensionais (estruturas) das imagens de forma invariante à translação, perspectiva e escala, assim como acontece no sistema de visão biológico.

A camada convolutiva é composta por um banco de filtros convolucionais (*kernels*), e para cada filtro é realizada a operação de convolução com a entrada 2-D, resultando em

mapas de características como saída, onde cada filtro convolvido com a imagem gera um mapa. A convolução de um filtro com uma imagem de entrada é ilustrada na figura 4.

Figura 4 – Calculando os valores de saída da convolução discreta [32]



Para esta operação há 4 parâmetros a serem definidos:

- Número de filtros (NF)
- Dimensão dos filtros (mxn)
- Avanço (S)
- Preenchimento com zero (P)

O número de filtros determina o número de mapas de características na saída da camada. Por exemplo, um camada convolucional com 3 filtros para uma entrada resulta em 3 mapas de características na saída. A dimensão do mapa de característica dependerá dos parâmetros adotados na camada e pode ser calculada por $(M - m)/S + 1$, para uma camada sem preenchimento com zero. O parâmetro S define o tamanho do avanço do filtro sobre a imagem e o preenchimento com zero amplia as bordas da imagem para que não haja

perda de dimensão da saída em relação à entrada. Para a figura 4, o avanço utilizado foi $S = 1$ e não houve preenchimento com zero. Assim como na MLP, os parâmetros listados acima também são definidos objetivando um melhor desempenho na tarefa executada pela RNA.

Duas importantes características desse tipo de camada podem ser evidenciadas: a redução do número dos parâmetros livres a serem ajustados e a extração robusta de formas bidimensionais da entrada. Onde a redução é devida ao compartilhamento dos pesos, pois um peso do filtro convolucional é compartilhado por diversos neurônios que recebem como sinal a imagem de entrada. De forma visual na figura 4, o peso de posição $(1, 1)$ do *kernel* multiplica-se aos pixels da entrada de posições $(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2) e (3, 3)$, diferentemente de uma MLP onde há um peso para cada entrada. Ademais, a extração robusta é obtida ao introduzir a operação de convolução com um pequeno kernel que ao deslizar sobre entrada extrai características invariantes a distorções.

Após uma camada convolutiva, uma rede convolutiva pode possuir uma camada de subamostragem, em que a dimensão do mapa de características é reduzida, com o efeito de reduzir a sensibilidade dos mapas a translações na entrada. Como o objetivo desse projeto está focado na camada convolutiva, a subamostragem não é desenvolvida.

Uma vez definida o tipo de rede a ser utilizado, seja ela CNN ou MLP, a implementação da mesma é finalizada com duas últimas etapas: o treinamento e, logo em seguida, o teste.

3.4 Treinamento e Teste

Com o intuito de obter um resposta classificando a entrada, o treinamento supervisonado consiste em apresentar à rede uma amostra de dados rotulados que representem o fenômeno real. É nesta etapa que ocorre o processo de aprendizagem e é importante ressaltar que o aprendizado torna-se mais eficiente à medida que é apresentado um conjunto numeroso de dados, que representa bem o universo do fenômeno. Uma boa maneira de garantir essa boa representação é apresentar bases de dados numerosas.

No contexto do processo de aprendizagem, há diversos métodos, sendo o método de aprendizagem por correção de erro mais comumente utilizado para treinar MLP e CNN, através do algoritmo de retropropagação de erro (em inglês, *error back-propagation*). Nesse caso, o erro é definido pela diferença entre a saída da rede e a classificação real (rótulo). A aprendizagem por retropropagação ocorre em duas etapas: o caminho para frente (algoritmo 1), onde os pesos sinápticos são fixos; e o caminho para trás, onde ocorre o ajuste dos pesos sinápticos de cada camada de acordo com uma regra de correção de erro. A regra mais utilizada consiste em definir uma função de custo composta do erro

(J) e ajustar os parâmetros de maneira a minimizar essa função, através da descida do gradiente. Por exemplo, considerando o algoritmo 1, o ajuste de $W1$, pode ser descrito na equação 3.3.

$$W1(n+1) = W1(n) - \eta \frac{\partial J}{\partial W1} \tag{3.3}$$

Onde n é o n-ésimo ajuste e η é a taxa de aprendizado.

Como o J depende de todos parâmetros livres da rede, a derivada parcial $\frac{\partial J}{\partial w1_{ij}}$ pode ser determinada através da regra da cadeia (equação 3.4), evidenciando ideia de retropropagação do erro até a primeira camada.

$$\frac{\partial J}{\partial W1} = \frac{\partial J}{\partial erro}\frac{\partial erro}{\partial Y}\frac{\partial Y}{\partial B}\frac{\partial B}{\partial D}\frac{\partial D}{\partial A}\frac{\partial A}{\partial W1} \tag{3.4}$$

O ajuste dos parâmetros é realizado diversas vezes em múltiplas apresentações de um conjunto de dados de treinamento (totalizando uma época) e pode ser feito de modo sequencial, onde os parâmetros são atualizados a cada apresentação de uma amostra à rede. Por fim, finalizado o treinamento (ajuste dos parâmetros), na etapa de teste são apresentados dados que não foram usados previamente. A saída na rede é então comparada com o rótulo real da amostra, e caso os valores coincidam em boa parte dos testes, o aprendizado é validado, atestando a capacidade de generalização da RNA.

3.5 Matrizes de transformação

Amplamente utilizadas em computação gráfica, as matrizes de transformações são a ferramenta de álgebra capazes de aplicar movimento aos elementos visuais. Dessa maneira, um objeto gráfico com descrição geométrica, em que seus vértices P são descritos em coordenadas no espaço 3D, são multiplicados à matrizes de transformação M para alterar sua localização de acordo com o movimento desejado. Esse objeto pode se mover através do espaço aplicando transformações geométrica: rotação, translação e escala. Neste projeto, são abordadas as duas primeiras transformações.

Uma matriz de rotação a partir da origem do sistema de coordenadas pode ser apresentada de acordo com o eixo de referência que se desejada rotacionar. As rotações em relação aos três eixos do espaço 3D estão descritas nas equações 3.5, 3.6 e 3.7.

$$M_{Rx}(\Theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\Theta & -sen\Theta \\ 0 & sen\Theta & cos\Theta \end{bmatrix} \tag{3.5}$$

$$M_{Ry}(\Theta) = \begin{bmatrix} cos\Theta & 0 & sen\Theta \\ 0 & 1 & 0 \\ -sen\Theta & 0 & cos\Theta \end{bmatrix} \tag{3.6}$$

$$M_{Rz}(\Theta) = \begin{bmatrix} cos\Theta & -sen\Theta & 0 \\ sen\Theta & cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

Como exemplo, para rotacionar com ângulo Θ em relação ao eixo x um objeto com vértices descritos em linhas de uma matriz P, as coordenadas dos vértices nas novas posições $P^{'}$ são dadas por $P^{'} = M_{Rx} \times P$.

Uma matriz de transformação afim T com coordenadas homogêneas e translação a partir da origem do sistema de coordenadas pode ser descrita pela equação 3.8, onde $t_x et_y$ são o deslocamento da posição do objeto em ralação aos eixos x e y, respectivamente.

$$M_T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

De forma resumida, dada uma matriz de transformação M e as coordenadas de um objeto P, pode-se gerar as novas coordenadas transformadas através de uma multiplicação $P^{'} = M \times P$.

Uma vez movido o objeto geométrico em 3D, é necessário projetá-lo em 2D e transformá-lo em matrizes de *bitmap* (imagens). Para isso, é preciso desenhar linhas entre os vértices de um objeto em formato de imagem. O algoritmo amplamente utilizado para desempenhar tal tarefa em computação gráfica, está descrito na seção a seguir.

3.6 Algoritmo para desenho de linhas

Uma das maneiras de desenhar um objeto geométrico em uma imagem de matriz de *pixels* consiste em selecionar os *pixels* (posições) na matriz da imagem para conectar dois pontos (vértices) do objeto, a fim de compor visualmente o objeto. Esta tarefa é chamada de rasterização por segmento de retas na área de computação gráfica. Existem diversos algoritmos para desenhar as linhas de um objeto, contudo, o algoritmo de Bresenham [33] é o mais utilizado na área, devido ao seu maior desempenho quanto à velocidade e precisão. Tal característica deve-se principalmente à utilização de número inteiros e da não utilização de divisões em sua realização.

Uma possível implementação do método está exposta no algoritmo 2, para desenho de reta entre dois pontos $(x_1, y_1)e(x_2, y_2)$, em que $x_2 > x_1$, com incremento em x e retorno de uma matriz de *pixels* definida pela rotina $preencherPixel(x, y)$, que atribui valor 1 à posição (x, y) da matriz .

Algorithm 2 Algoritmo de Bresenham

```
dy ← y2 − y1
dx ← x2 − x1                                  ▷ para valores onde x2 > x1
P ← 2 ∗ dy − dx
x ← x1
y ← y1
while x ⩽ x2 do
    preencherPixel(x, y)
    x ← x + 1
    if P < 0 then
        P ← P + 2 × dy
    else
        P ← P + 2 × dy − 2 × dx
        y ← y + 1
    end if
end while
```

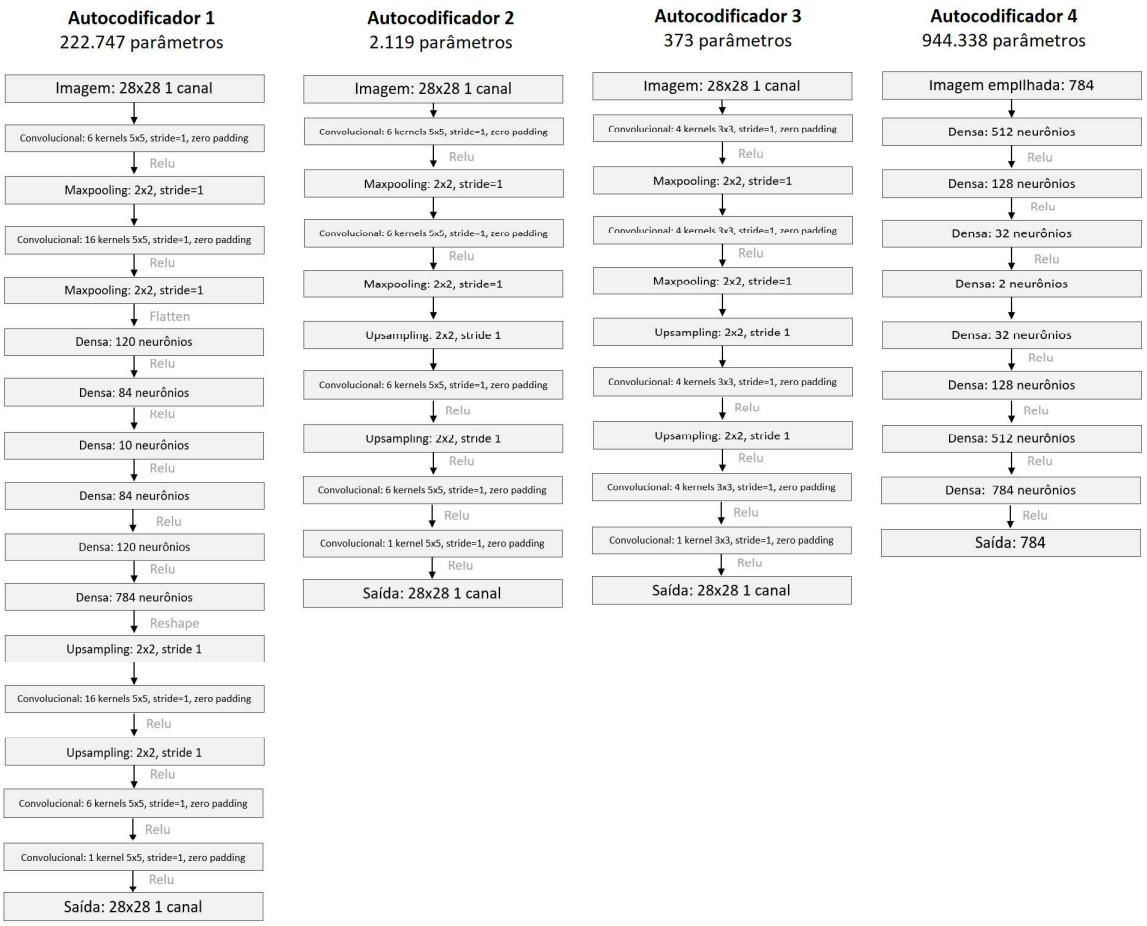
4 ABORDAGEM PROPOSTA

Como primeira etapa desenvolvida, foi realizada a revisão bibliográfica e o estudo teórico referentes aos conceitos e métodos implementados ao longo dos experimentos, descritos nos primeiros capítulos deste relatório. Após fundamentação da base teórica necessária para iniciar a pesquisa, com o intuito de implementar arquiteturas de redes mais complexas durante os experimentos, a abordagem de programação linha por linha em *Matlab*, adotada no projeto anterior, foi substituída pelo uso da biblioteca pública Keras[34] em linguagem *python*. Por ser o primeiro contato da autora deste projeto com a linguagem python, os primeiros autocodificadores implementados foram aplicados à base pública bastante utilizada MNIST [35]. Posteriormente, após a familiarização da ferramenta com o uso de base clássica, os autocodificadores foram aplicados às imagens de representações icônicas.

Para aquisição de domínio da linguagem, foram implementados 4 arquiteturas de autocodificadores. A descrição das camadas destas redes estão expostas na figura 5.

O primeiro autocodificador possui arquitetura de codificação baseada na LeNet [6]. A segunda arquitetura consiste em uma rede puramente convolutiva. Já a terceira possui quase a mesma organização, porém com número menor de parâmetros, ou seja, é uma

Figura 5 – *Autocodificadores implementados para MNIST*



rede mais restrita. O último autocodificador é de arquitetura densa, a fim de comparar seu desempenho com as outras arquiteturas convolutivas já implementadas.

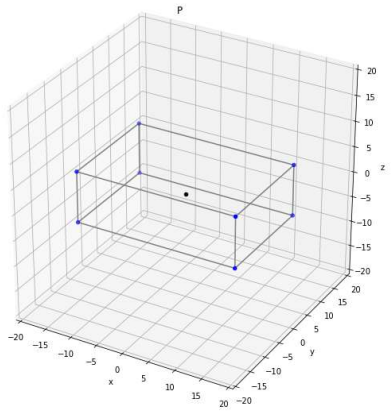
Após a aplicação de arquiteturas à base pública, iniciou-se a construção de uma base sintética formada por ícones. É importante ressaltar que este estudo está sendo realizado inicialmente em bases sintéticas formada por representação icônica de um objeto rígido, para futuramente aplicá-lo aos equipamentos presentes nas plantas elétricas da termelétrica. Dessa maneira, a figura 6 ilustra a posição e orientação de uma estrutura rígida simplificada do equipamento.

O ícone padrão foi definido em espaço vetorial 3D, e trata-se de uma paralelepípedo de comprimento 30, largura 20 e altura 10 (em dimensões arbitrárias), exposto na figura 7. Para cada amostra a ser gerada, cinco transformações são submetidas ao prisma padrão: três rotações em cada eixo eixo x, eixo y e eixo z) e duas translações em cada eixo (eixo x e eixo y). As transformações referentes à rotação (orientação) da representação icônica foram geradas aleatoriamente com distribuição uniforme entre 0 e 2π. As transformações referentes à translação (posição) foram geradas aleatoriamente com distribuição uniforme

Figura 6 – *Representação icônica de objetos rígidos*

entre -30 e 30. Tais valores foram definidos para que o desenho do prisma mantenha-se dentro da matriz definida de *pixels*.

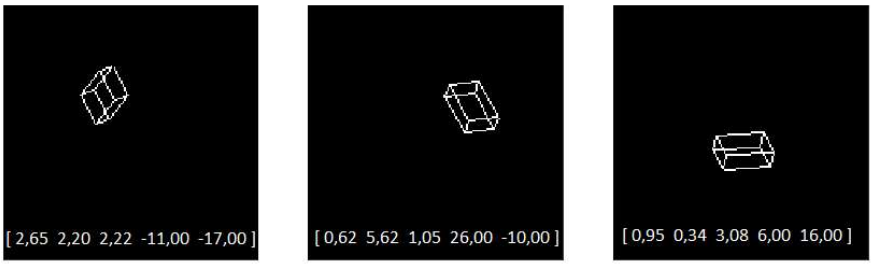
Figura 7 – *Representação icônica padrão*



Em seguida, antes de gerar a imagem em matriz de pixels, o ícone foi projetado em 2D, sem perspectiva de profundidade. Logo, translações em z são imperceptíveis à projeção e portanto não fazem parte das transformações aplicadas anteriormente. Para gerar a imagem da projeção foi implementado o algoritmo de desenho de linha de *Bresenham*[33] entre os pontos da projeção em duas dimensões. Ao todo, foram geradas 15000 amostras de imagens de dimensão 150x150 de *pixels* do ícone, cada amostra associada a suas respectivas cinco variáveis de transformação (rotação/translação), para treinamento das redes, e 1000 amostras para teste. A figura 8 expões 3 exemplos de amostras e suas respectivas variáveis de transformação Z, sendo as 3 primeiras de orientação (em radianos) e as 2 últimas de posição (em número de *pixels*).

A construção da base icônica gerou, através de técnicas de computação gráfica, uma associação de variáveis de transformação Z a suas respectivas imagens I do prisma transformado, podendo ser representado por uma função $g : \varphi \rightarrow \chi$ que mapeia as transformações Z contidas no espaço φ em imagens I contidas no espaço de poses possíveis do prisma χ.

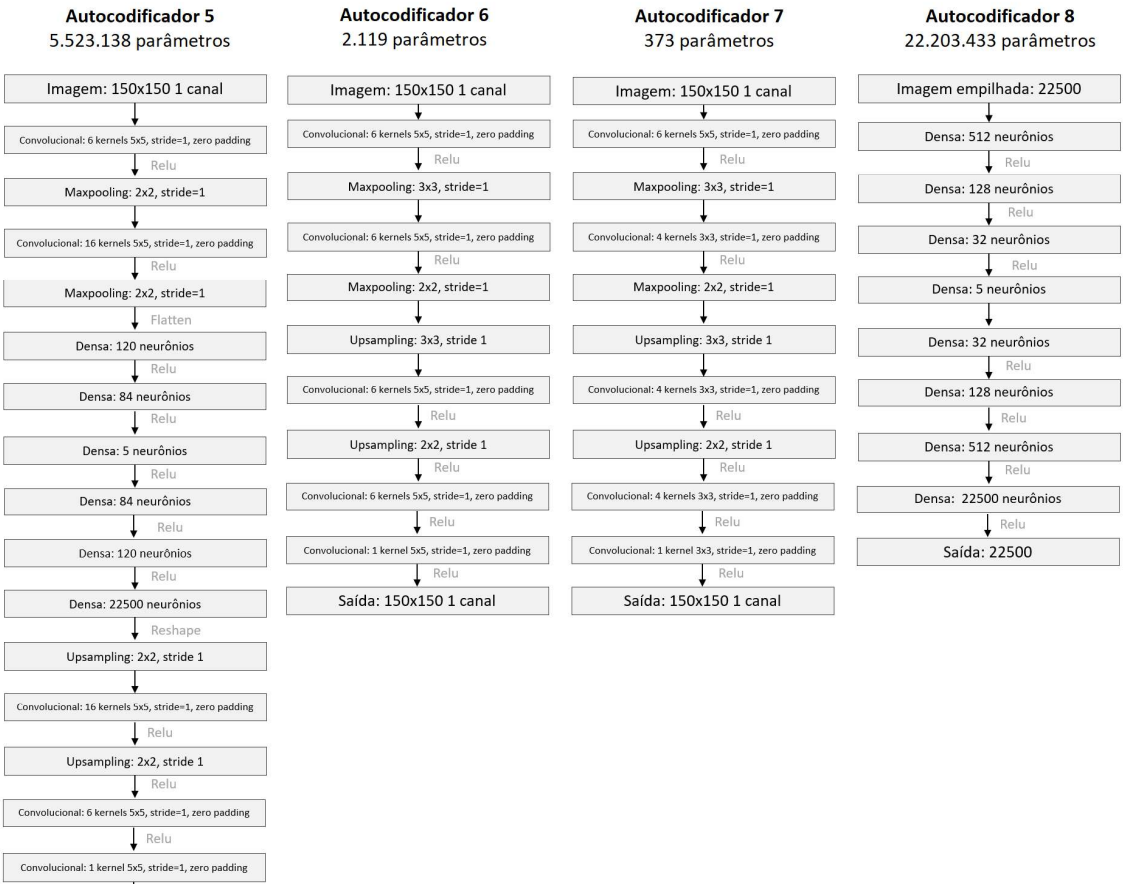
Figura 8 – *3 amostras da Base de Representação icônica*



Seguidamente, foi realizado um estudo para aprendizado explícito e controlado das variáveis de transformação associadas a cada amostra. Em outras palavras, foi investigada a capacidade de autocodificadores, de redes neurais MLP e ConvNets mapearem o caminho inverso da sintetização $g^{-1} : \chi \rightarrow \varphi$, determinando a partir das imagens I suas respectivas variáveis de transformação Z, ou seja, determinando a posição e orientação da representação icônica.

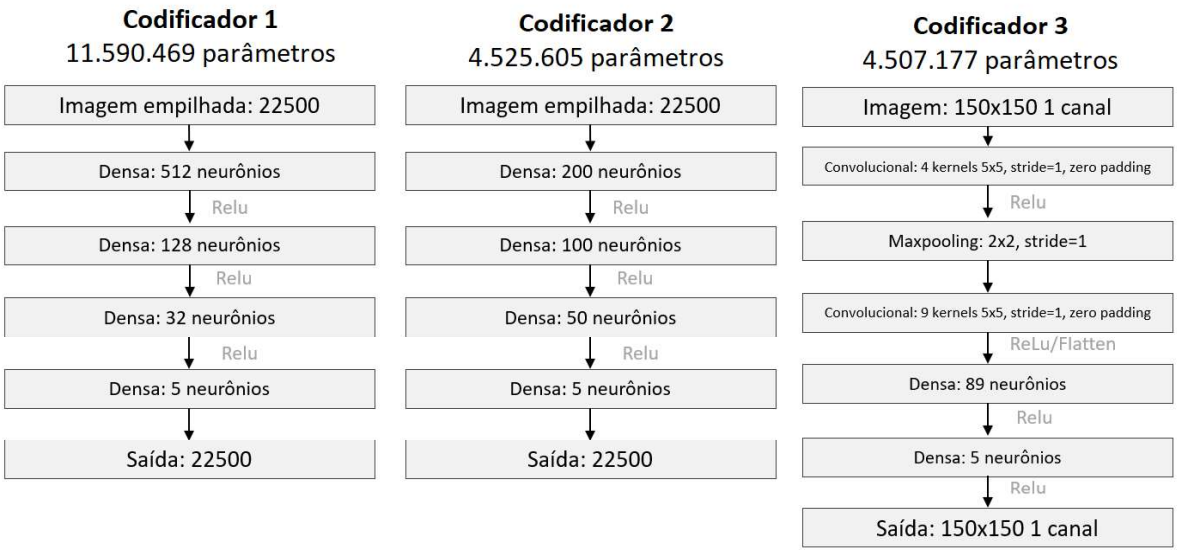
As arquiteturas de autocodificadores implementadas para aprendizado não supervisionado estão expostas na figura 9, seguindo o mesmo padrão das configurações adotadas para base MNIST.

Figura 9 – *Autocodificadores implementados para base de representações icônicas*



Para encontrar a possível representação desejada durante a codificação da rede, dividiu-se a aplicação e estudo do autocodificador em duas estruturas independentes: codificador e decodificador. Desta maneira, separada da estrutura de um autocodificador clássico, a codificação foi realizada de maneira supervisionada, onde as saídas do codificador desejadas são as transformações representadas por Z. Sendo assim, força-se que a entrada seja mapeada no espaço de dimensão reduzida desejado φ, ou seja, o espaço latente onde está o *manifold* referente à posição/orientação do ícone na imagem. As arquiteturas dos codificadores implementados estão expostas na figura 10. A primeira arquitetura tem configuração idêntica ao codificador do autocodificador 8 (figura 9). O codificador 2 possui de uma arquitetura densa, porém mais restrita. E, para fins comparativos, a terceira arquitetura é convolutiva com aproximadamente mesma dimensão da segunda.

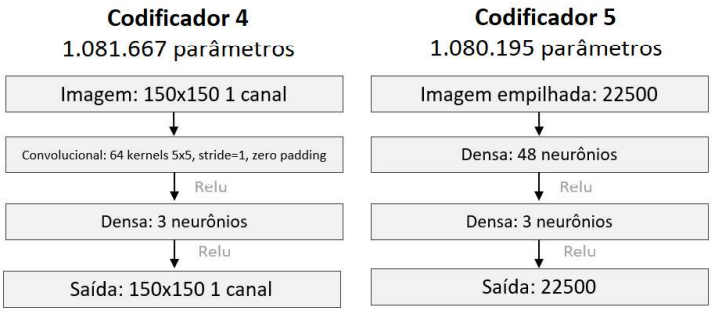
Figura 10 – *Codificadores implementados para base de representações icônicas*



A fim de reduzir a quantidade de possibilidades de movimentação do ícone padrão (7) e, consequentemente, facilitar o mapeamento desempenhado pela rede, criou-se uma nova base de ícones com 4 possibilidades de rotação e sem possibilidade de translação. Assim sendo, a base possui 64 poses possíveis do prisma na imagem para treinamento e 1000 poses de qualquer rotação para teste. Novos experimentos de codificação foram realizados por redes com arquiteturas descritas na figura 11.

Por último, com o intuito de investigar as formas das 64 poses presentes na nova base, foi implementado um método simples e direto, intitulado de "método linha de base". Esse experimento determinou o produto escalar entre as 1000 imagens de teste $X_i = (x_1, x_2, ..., x_d)$, onde $i = 1, 2, ..., 1000$, e as 64 imagens de referência $R_j = (r_1, r_2, ..., r_d)$, onde $j = 1, 2, ..., 64$, tendo como saída uma matriz DP de dimensão 1000x64 determinada

Figura 11 – *Codificadores implementados para base de representações icônicas de 64 poses*



pela equação 4.1. É importante sublinhar que cada imagem foi representada na forma de um vetor de norma unitária.

$$DP_{ij} = \sum_{k=1}^{d} x_k r_k \tag{4.1}$$

Os resultados obtidos ao longo do desenvolvimento do projeto de acordo com a abordagem proposta descrita nesta sessão estão descritos no capítulo a seguir.

5 RESULTADOS

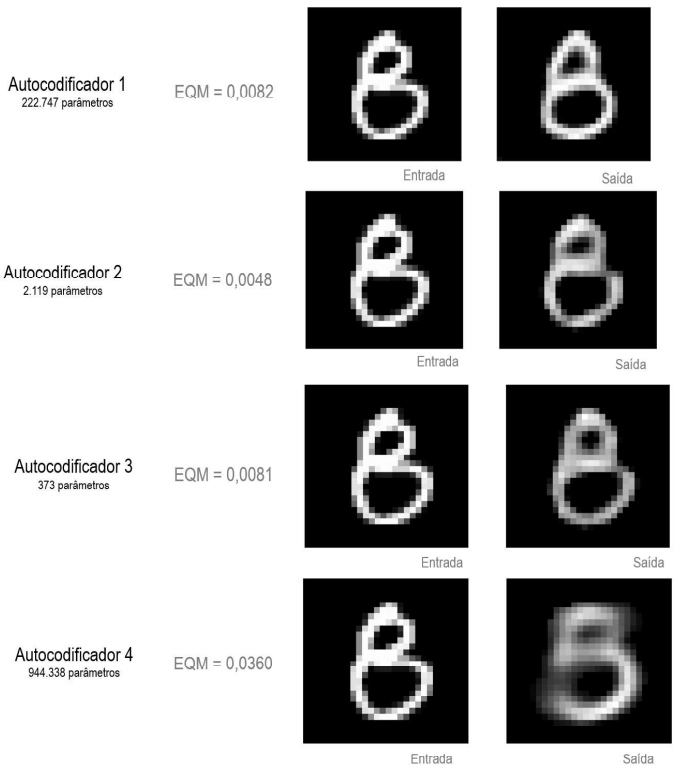
Para a implementação inicial dos 4 primeiros autocodificadores descritos no capítulo anterior (figura 5) e com a base MNIST como entrada, as imagens de reconstrução obtidas e o erro quadrático médio (EQM) na saída para fase de teste estão expostos na figura 12.

Pode-se perceber que o autocodificador 2 puramente convolutivo e mais restrito apresentou menor EQM quando comparado ao autocodificador 1, baseado na codificação da LeNet[6]. Outro resultado relevante foi o aumento do EQM e piora na reconstrução para o autocodificador 4 denso, ratificando o melhor desempenho de redes convolutivas quando aplicadas a imagens.

Uma vez finalizada a experimentação com base pública, os novos experimentos, realizados inicialmente para quatro arquiteturas (figura 9) utilizando a base de representações icônicas como entradas, apresentaram os resultados expostos na figura 13.

Assim como nos resultados apresentados pela base MNIST, os autocodificadores convolutivos (5,6 e 7) apresentaram melhor reconstrução da entrada na saída, confirmando o aprendizado da forma e posição do prisma na imagem. O autocodifcador denso foi capaz de reconstruir a posição, porém a forma, onde está inserida a informação de orientação, ficou mais prejudicada.

Figura 12 – *Saída dos Autocodificadores implementados para base de MNIST*

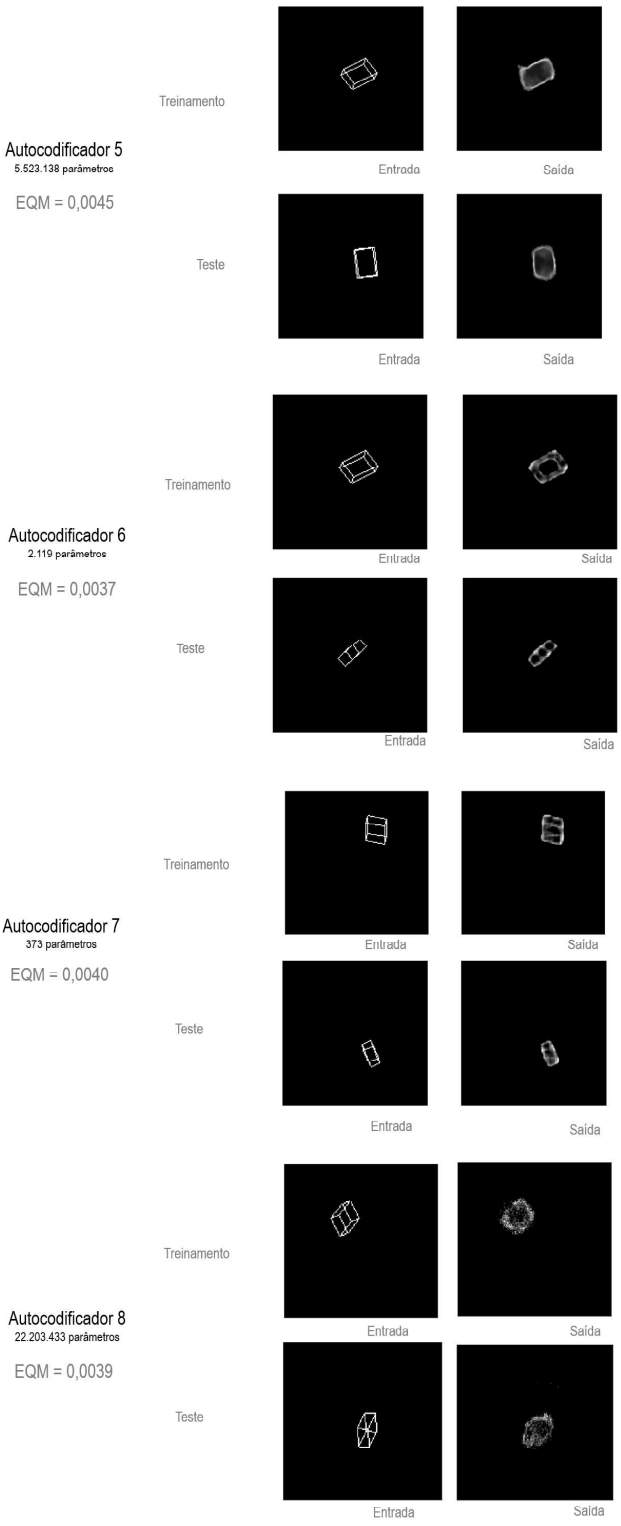


Dando atenção ao estudo explícito de variáveis latentes, a saída de dimensão reduzida (gargalo da estrutura) para o autocodificador 8 foi avaliada. Nesta arquitetura implementada, a saída H de cinco dimensões no gargalo da rede foi comparada à variável latente desejada Z, onde Z contém os cinco parâmetros utilizadas na geração das amostras da base de ícones. Durante experimento comparativo, verificou-se que H não pode ser representado como uma transformação linear de Z, confirmando um possível mapeamento não-linear entre entrada e gargalo. Além disso, foi observado que, para as estruturas testadas, com até 22,2 milhões de parâmetros livres a serem ajustados/treinados, H não apresentou variáveis independentes entre si. Sendo assim, por treinamento clássico (não supervisionado), o autocodificador foi incapaz de mapear, em seu codificador, as variáveis latentes de um *manifold* que poderiam representar a informação de translação e rotação do ícone.

Para a tarefa de codificação supervisionada, as arquiteturas implementadas (figura 10) apresentaram os resultados expostos na tabela 1.

Para o treinamento, o codificador convolutivo (codificador 3) apresentou um menor EQM. Este resultado é esperado devido à superioridade conhecida desse tipo de rede em aplicação com imagens. Porém, experimentos de teste realizados com as 3 arquiteturas indicaram problemas de regularização, mesmo restringindo o número de parâmetros livres e alterando o tipo da rede.

Figura 13 – *Saída dos Autocodificadores implementados para base de representações icônicas*



Aplicando-se a base de menores possibilidades de rotação e sem translação, os resultados apresentados para dois grupos de ângulos possíveis e dois codificadores implementados (figura 11) estão expostos na tabela 2.

Tabela 1 – EQM na saída dos codificadores implementados para base de representações icônicas

	Codificador 1	Codificador 2	Codificador 3
EQM na saída para treinamento	$4,23.10^{-4}$	$4,04.10^{-4}$	$1,16.10^{-4}$
	$2,68.10^{-4}$	$5,85.10^{-4}$	$1,15.10^{-4}$
	$4,49.10^{-4}$	$4,19.10^{-4}$	$1,00.10^{-4}$
	$9,71.10^{-5}$	$1,14.10^{-4}$	$1,28.10^{-5}$
	$6,36.10^{-5}$	$9,10.10^{-5}$	$1,54.10^{-5}$
EQM na saída para teste	$9,67.10^{-2}$	$1,04.10^{-1}$	$1,18.10^{-1}$
	$1,01.10^{-1}$	$1,08.10^{-1}$	$1,25.10^{-1}$
	$8,98.10^{-2}$	$9,15.10^{-2}$	$1,05.10^{-1}$
	$1,81.10^{-3}$	$2,48.10^{-3}$	$2,29.10^{-3}$
	$1,40.10^{-3}$	$2,50.10^{-3}$	$2,20.10^{-3}$

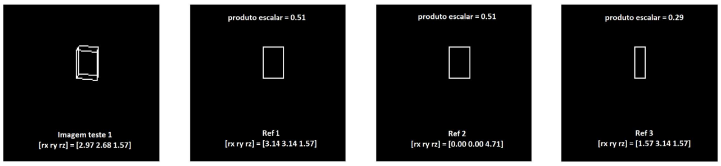
Tabela 2 – EQM na saída dos codificadores implementados para base de representações icônicas com 64 poses

		Codificador 4	Codificador 5
EQM na saída para 45° 110° 250° 330°	treinamento	$\cong 0$	$\cong 0$
		$\cong 0$	$\cong 0$
		$\cong 0$	$\cong 0$
	teste	$0,11$	$0,13$
		$0,12$	$0,15$
		$0,09$	$0,09$
EQM na saída para 0° 90° 180° 270°	treinamento	$0,07$	$0,07$
		$0,06$	$0,06$
		$0,07$	$0,07$
	teste	$0,10$	$0,19$
		$0,11$	$0,24$
		$0,10$	$0,18$

A incapacidade de generalização das redes permaneceu. Porém, um resultado inusitado obtido nesse experimento foi o erro de treinamento para o primeiro grupo de ângulos convergindo para aproximadamente zero e, por outro lado, para mesmas arquiteturas, mesmas limitações e somente o grupo de possibilidades diferente, o erro de treinamento obtido foi alto. Tal ocorrência inspirou o último experimento deste estudo intitulado "linha de base", descrito no capítulo anterior.

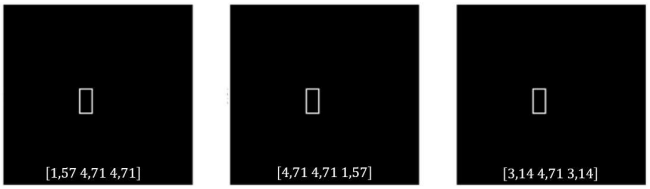
Ao longo da análise dos produtos escalares entre imagens de teste e referências, os valores obtidos evidenciaram que a mesma imagem teste possuía mesmo resultado de produto escalar para diferentes variáveis de transformação associadas às imagens de referência. O valor dos três maiores, escalares de uma das amostras da base teste aplicados ao segundo grupo de ângulos, está exposto na figura 14.

Figura 14 – *Método Linha de Base*



Foi percebido neste resultado uma característica importante da base de 64 poses para ângulos 0° 90° 180° 270°(em radianos, 0rad 1,57rad 3,14rad 4,71rad): diferentes variáveis de transformação geram, por simetria do prisma, a mesma projeção na imagem. Sendo assim, há ambiguidades presentes na base onde diferentes valores Z estão associados a uma mesma imagem I (figura 15).

Figura 15 – *Ambiguidade da base de 64 poses para ângulos 0° 90° 180° 270°*



Ao observar essa característica, pôde-se concluir que uma possível causa para o baixo desempenho de aprendizado das redes aplicadas neste estudo reside na incapacidade da rede mapear as relações de ambiguidade presentes nas bases de representações icônicas, uma vez que, redes neurais determinísticas (e.g MLP e ConvNets) relacionam uma entrada I em uma saída Z através de funções matemáticas, e por definição, cada elemento do conjunto I precisa ter um único correspondente no conjunto Z.

As conclusões deste projeto e projeções para trabalhos futuros estão descritas no capítulo a seguir.

6 CONCLUSÃO

A implementação utilizando a biblioteca Keras[34] garantiu uma rapidez na implementação de redes neurais adotadas, permitindo a realização de numerosas arquiteturas e diversos experimentos.

A criação de uma base de representações icônicas proporcionou uma aprendizado explícito das variáveis latentes desejadas. Ao criar uma imagem com apenas um objeto simples a partir de transformações (rotações e translações) e utiliza-las para aprendizado das redes, pode-se controlar explicitamente o que a rede deve aprender, em outras palavras,

em que a rede deve mapear o objeto de entrada. Cabe notar que esse estudo trata dos fundamentos que devem conduzir ao resultado prático da segmentação e estimação automática posição e rotação de um objeto em uma imagem digitalizada.

No âmbito da análise de posição/orientação como forma avançada na classificação de objetos, e em conjunto com a ideia proposta de particionar a tarefa de codificação/decodificação de um autocodificador, apesar das ConvNets terem apresentado um menor erro de treinamento ambas arquiteturas ainda não apresentaram bons resultados para generalização. Por consequência, a investigação de um método mais simples e direto que uma rede neural evidenciou o problema de relação ambígua. Portanto, de uma maneira geral, a posição/orientação de um objeto com simetria não pode ser estimada, seja o objeto um simples ícone ou até mesmo a imagem detalhada de um equipamento elétrico, por uma rede neural determinística, não importando tamanho ou estrutura dessa rede.

Com o intuito de promover andamento à pesquisa para desenvolvimento de um método capaz de estimar de maneira automática posição/orientação de uma representação icônica e futuramente aplicá-lo em ícones mais próximos ao contorno do equipamento, os trabalhos futuros devem transcorrer no sentido da utilização de modelos probabilísticos capazes de associar uma mesma imagem a prováveis diferentes pontos de vista. Ademais, objetivando uma possível inspiração para criação do novo projeto, será feito um estudo e revisão sobre métodos para estimação de ponto de vista (orientação e posição), presentes na literatura de estado da arte, principalmente no que diz respeito à ambiguidade de poses.

Referências

[1] LEI, Y. et al. An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data. *IEEE Transactions on Industrial Electronics*, v. 63, n. 5, p. 3137–3147, 2016.

[2] WANG, H.; ZHOU, B.; ZHANG, X. Research on the remote maintenance system architecture for the rapid development of smart substation in china. *IEEE Transactions on Power Delivery*, v. 33, n. 4, p. 1845–1852, 2018.

[3] SALEM, A. A. et al. The leakage current components as a diagnostic tool to estimate contamination level on high voltage insulators. *IEEE Access*, v. 8, p. 92514–92528, 2020.

[4] WERNECK, M. M. et al. Detection and monitoring of leakage currents in power transmission insulators. *IEEE Sensors Journal*, v. 15, n. 3, p. 1338–1346, 2015.

[5] CUN, Y. L. et al. Handwritten zip code recognition with multilayer networks. In: IEEE. *[1990] Proceedings. 10th International Conference on Pattern Recognition*. [S.l.], 1990. v. 2, p. 35–40.

[6] LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, 1998.

[7] HEARST, M. et al. Support vector machines. *IEEE Intelligent Systems and their Applications*, v. 13, n. 4, p. 18–28, 1998.

[8] REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 39, n. 6, p. 1137–1149, 2017.

[9] DOSOVITSKIY, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. Disponível em: <https://arxiv.org/abs/2010.11929>.

[10] MA, Y.; FU, Y. *Manifold Learning Theory and Applications*. [S.l.]: CRC Press, 2012.

[11] JOLLIFFE, I. *Principal Component Analysis*. [S.l.]: Springer Verlag, 1986.

[12] KRUSKAL, J. B. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, Springer, v. 29, n. 1, p. 1–27, 1964.

[13] TENENBAUM, J. B.; SILVA, V. d.; LANGFORD, J. C. A global geometric framework for nonlinear dimensionality reduction. *science*, American Association for the Advancement of Science, v. 290, n. 5500, p. 2319–2323, 2000.

[14] ROWEIS, S. T.; SAUL, L. K. Nonlinear dimensionality reduction by locally linear embedding. *science*, American Association for the Advancement of Science, v. 290, n. 5500, p. 2323–2326, 2000.

[15] RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by error propagation. In: _____. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986. p. 318–362. ISBN 026268053X.

[16] BANK, D.; KOENIGSTEIN, N.; GIRYES, R. Autoencoders. *arXiv preprint arXiv:2003.05991*, 2020.

[17] RANZATO, M. et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2007. p. 1–8.

[18] PU, Y. et al. Variational autoencoder for deep learning of images, labels and captions. In: LEE, D. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2016. v. 29. Disponível em: <https://proceedings.neurips.cc-/paper/2016/file/eb86d510361fc23b59f18c1bc9802cc6-Paper.pdf>.

[19] MASCI, J. et al. Stacked convolutional auto-encoders for hierarchical feature extraction. In: SPRINGER. *International conference on artificial neural networks*. [S.l.], 2011. p. 52–59.

[20] FUKUSHIMA, K.; MIYAKE, S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In: *Competition and cooperation in neural nets*. [S.l.]: Springer, 1982. p. 267–285.

[21] KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012. v. 25. Disponível em: <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>.

[22] SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] SZEGEDY, C. et al. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 1–9.

[24] HE, K. et al. Deep residual learning for image recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. p. 770–778.

[25] KAYALIBAY, B.; JENSEN, G.; SMAGT, P. van der. Cnn-based segmentation of medical imaging data. *CoRR*, abs/1701.03056, 2017. Disponível em: <http://arxiv.org-/abs/1701.03056>http://arxiv.org/abs/1701.03056.

[26] XIONG, X. et al. Identification of electrical equipment based on faster lstm-cnn network. In: *2020 IEEE International Conference on Networking, Sensing and Control (ICNSC)*. [S.l.: s.n.], 2020. p. 1–6.

[27] LEI, X.; SUI, Z. Intelligent fault detection of high voltage line based on the faster r-cnn. *Measurement*, Elsevier, v. 138, p. 379–385, 2019.

[28] JIANLONG, G.; WEIXIA, F.; MANHUA, W. An improved faster r-cnn algorithm for electric equipment detection. In: *Proceedings of the 2019 11th International Conference on Computer and Automation Engineering*. [S.l.: s.n.], 2019. p. 138–141.

[29] HAN, S. et al. Electrical equipment identification in infrared images based on roi-selected cnn method. *Electric Power Systems Research*, Elsevier, v. 188, p. 106534, 2020.

[30] HAYKIN, S. *Redes Neurais - 2ed.* Bookman, 2001. ISBN 9788573077186. Disponível em: <https://books.google.com.br/books?id=lBp0X5qfyjUC>.

[31] HUBEL, D. H.; WIESEL, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, Wiley-Blackwell, v. 160, n. 1, p. 106, 1962.

[32] DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[33] BRESENHAM, J. E. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, v. 4, n. 1, p. 25–30, 1965.

[34] CHOLLET, F. et al. *Keras*. 2015. `https://keras.io`.

[35] LECUN Y., C. C.; BURGES, C. The mnist database of handwritten digits. 1998. Disponível em: <\urlhttp://yann.lecun.com/exdb/mnist>.