Matheus Aparecido do Carmo Alves – 9791114

Projeto 1: Análise, Modelagem e Classificação dos conjuntos de dados Iris e MNIST

Professor Doutor Bruno Squizato Faiçal

SCC0270 Redes Neurais e Aprendizado Profundo

Instituto de Ciências Matemáticas e de Computação - ICMC

Universidade de São Paulo - USP

Bacharelado em Ciências de Computação

São Carlos, SP

30 de abril de 2020

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Este documento refere-se ao projeto número 1 da disciplina Redes Neurais e Aprendizado Profundo (SCC0270) oferecido pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC / USP) [1].

Esta primeira seção visa realizar uma breve introdução, revelando o foco do projeto. Assim, uma contextualização sobre o assunto principal a ser desenvolvido neste documento será feita na Seção 1.1. Em seguida, serão apresentados os motivadores associados ao desenvolvimento deste trabalho na Seção 1.2. Na Seção 1.3 serão apresentados os objetivos do projeto, retirados do documento de especificação. E por fim, será apresentada a organização deste documento na Seção 1.4.

1.1 Contextualização

Desde a década de 40, estudos a cerca da aplicação de redes neurais para resolução de problemas são desenvolvidos na tentativa de viabilizar o processamento de dados por meio de um algoritmo que permita o aprendizado de padrões sem a necessidade uma rigorosa formulação matemática para a criação do modelo. Apresentada juntamente ao desenvolvimento de diversos paradigmas, as redes neurais evoluíram até o ponto de possibilitarem o processamento e classificação de grandes bases de dados eficientemente.

Para tal, arquiteturas cada vez mais complexas foram estabelecidas e aprimoradas para generalizar o conhecimento obtido durante o processo de treinamento da rede. A formulação e proposta do aprendizado profundo ofereceu as redes neurais uma nova forma de se melhorar a aquisição de conhecimento de uma maneira rápida e simples, deixando como principal desafio à comunidade o ajuste dos parâmetros das arquiteturas propostas que melhor se encaixe no cenário de estudo.

Desta forma, este projeto visa abordar este desafio na resolução de problemas bem definidos da literatura: classificação do conjunto de dados *Iris* e *MNIST*. O *dataset Iris* representa um conjunto de dados sobre plantas, as quais são classificadas segundo certos parâmetros métricos que descrevem seus formatos. Já o *dataset MNIST* apresenta um conjunto de imagens que apresentam números naturais de 0 à 9 escritos a mão. Ambos conjuntos de dados serão melhores apresentados na Seção 2, assim como a abordagem escolhida e arquitetura para resolução dos problemas serão apresentados na Seção 3.

[1] Todos os códigos referentes a disciplina são públicos e estão disponíveis no GitHub/Micanga.

1.2 Motivação

O uso de algoritmos de aprendizado via Redes Neurais vem se tornando algo cada vez mais comum e poderoso no contexto de desenvolvimento de tecnologia e resolução de problemas para a computação atual. Seja por grandes empresas ou em aplicações acadêmicas, as redes neurais e técnicas de aprendizado profundo já mostraram capacidade para o processamento de grande base de dados, imagens, jogos, assim como para desenvolvimento de inteligência embarcada (1, 2, 3, 4). Assim, esta área de estudo tem ganhado importância para o ensino e aprendizado dentro da universidade e em empresas, visando melhorar técnicas e a aquisição de conhecimento por meio da análise de dados.

Este projeto visa demonstrar esta importância, corroborando para o aprendizado e desenvolvimento de conhecimento acerca do assunto; importante quanto alunos de ciências da computação e futuros profissionais na área. A diversa aplicação, facilidade de desenvolvimento e eficiência das redes neurais ilustra a extrema importância de seu conhecimento para atuação na área, justificando e motivando a realização deste projeto.

1.3 Objetivos

Os objetivos definidos para a realização deste projeto contemplam:

1. ***Dataset Íris***

   a) a implementação dois classificadores MLP com arquiteturas distintas de modo a tentar se alcançar a melhor classificação possível somente pelo ajuste de seus parâmetros;

   b) a realização de uma análise para validação do classificador e avaliação do desempenho sobre uma métrica padrão, e;

   c) o desenvolvimento de discussão sobre a taxa de generalização alcançada, apontando se está foi satisfatória ou não com base nos resultados obtidos.

2. ***Dataset MNIST*:**

   a) a implementação dois classificadores: um classificador Convolucional e um classificador *Support Vector Machine*-Linear (SVM-Linear).

   b) o desenvolvimento de uma análise para validação do classificador e avaliação do desempenho sobre uma métrica padrão;

   c) a discussão sobre a taxa de generalização alcançada, apontando se está foi satisfatória ou não (com base nos resultados obtidos), e;

   d) a apresentação de uma comparação do resultado alcançado com métodos catalogados no site oficial do conjunto de dados MNIST.

1.4 Organização

Desta forma, este documento está organizado da seguinte maneira: A Seção 2 irá apresentar e detalhar ambos conjuntos de dados a serem explorados neste projeto. A Seção 3 descreverá as arquiteturas e métodos implementados para resolução dos dois problemas propostos. A Seção 4 será responsável por conduzir as discussões a cerca dos resultados obtidos, testes e métricas utilizadas. E por fim, a Seção 5 apresentará a conclusão alcançada ao fim do desenvolvimento do trabalho.

2 Descrição da base de dados

Esta seção tem por objetivo realizar a apresentação dos conjuntos de dados a serem analisados dentro deste projeto. Desta forma, primeiramente será apresentado o conjunto de dados *Íris* na Seção 2.1, descrevendo os parâmetros, número de dados, organização e as classes disponíveis. Em um segundo momento, será descrito o conjunto de dados *MNIST* na Seção 2.2 apresentando parte das imagens e características a serem analisadas.

2.1 Dataset: Íris

O conjunto de dados *Íris* é, originalmente, apresentado em um arquivo *.csv* que contem seus dados separados por "virgulas" (","); organizados em uma tabela. Esta tabela contém 5 colunas e 150 linhas. Portanto, o conjunto de dados apresenta 5 variáveis, sendo 4 colunas para parâmetros e 1 coluna para a informação de classe para um total de 150 exemplos (ou instâncias).

Assim, as variáveis do conjunto de dados são:

- *sepal_length*: comprimento da sépala em centímetros.
- *sepal_width*: largura da sépala em centímetros.
- *petal_length* comprimento da pétala em centímetros.
- *petal_width*: largura da pétala em centímetros.
- *class*: classe da instância, podendo ser: (0) Iris Setosa, (1) Versicolor ou (2) Virgínica.

A Figura 1 apresenta um *DataFrame* produzido pela biblioteca *Pandas* que mostra parte dos dados contidos no conjunto de dados e sua organização.

Analisando a estratificação dos dados de acordo com as classes disponíveis, observa-se que o conjunto de dados é balanceado, contendo 50 amostras da classe 0 (setosa), 50 amostras da classe 1 (versicolor) e 50 amostras da classe 2 (virgínica). Esta informação é importante, uma vez que delimita abordagens possíveis para se definir métrica de teste, treino e avaliação de generalização de conhecimento.

Neste projeto, o conjunto de dados foi carregado através do método *load_iris()* da biblioteca *sklearn.datasets*, disponível para a linguagem *Python*.

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2.0
146	6.3	2.5	5.0	1.9	2.0
147	6.5	3.0	5.2	2.0	2.0
148	6.2	3.4	5.4	2.3	2.0
149	5.9	3.0	5.1	1.8	2.0

150 rows × 5 columns

Figura 1 – *DataFrame* do conjunto de dados *Íris* gerado pala biblioteca Pandas.

2.2 Dataset: MNIST

O conjunto de dados MNIST constitui um conjunto de dados de digitos escritos a mão disponível online publicamente. O conjunto possui 60,000 exemplos já separados para treinamento e 10,000 exemplos para teste. Todos os dígitos possuem seu tamanho normalizado (com dimensão 28x28) e estão centralizados no arquivo de imagem. Geralmente utilizado para teste base de métodos de aprendizado e reconhecimento de padrões, o conjunto MNIST representa uma importante base para a comunidade no desenvolvimento de técnicas associadas a este contexto. A Figura 2 apresenta exemplos de imagens que compõe o conjunto.

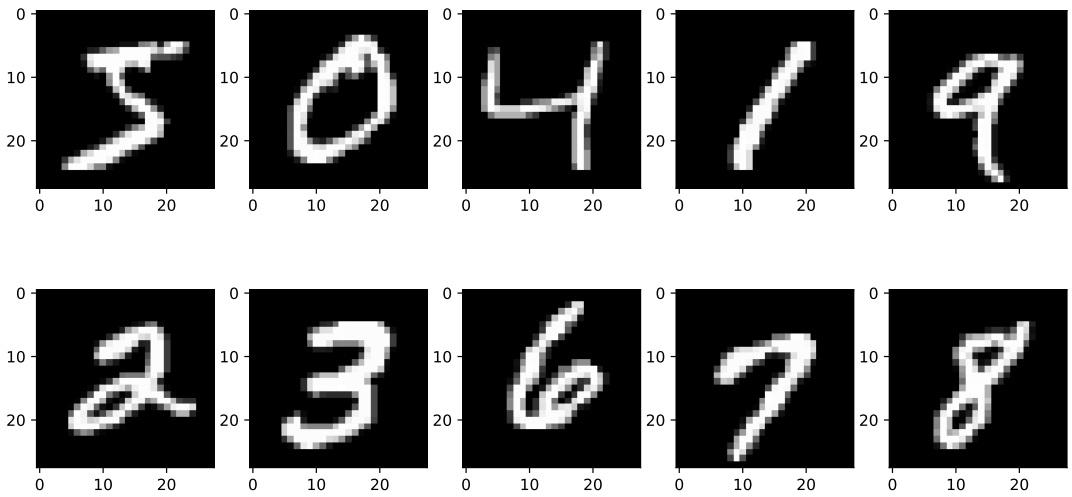

Figura 2 – Apresentação de um exemplo de cada classe disponível no conjunto de dados *MNIST*. Os números nas imagens são $5, 0, 4, 1, 9, 2, 3, 6, 7$ e 8, respectivamente.

Desta forma, é possível elencar as características deste conjunto de dados que possui:

- 784 parâmetros por imagem (número total de pixels da imagem 28x28);
- 10 possíveis classes (dígitos – 0 à 9);
- um total de 70,000 exemplos.

A distribuição de classes do conjunto é desbalanceada, i.e., o número de amostras para cada classe no conjunto de dados não é o mesmo. A Figura 3 apresenta a quantidade percentual de amostras para cada uma das 10 possíveis classes no conjunto.

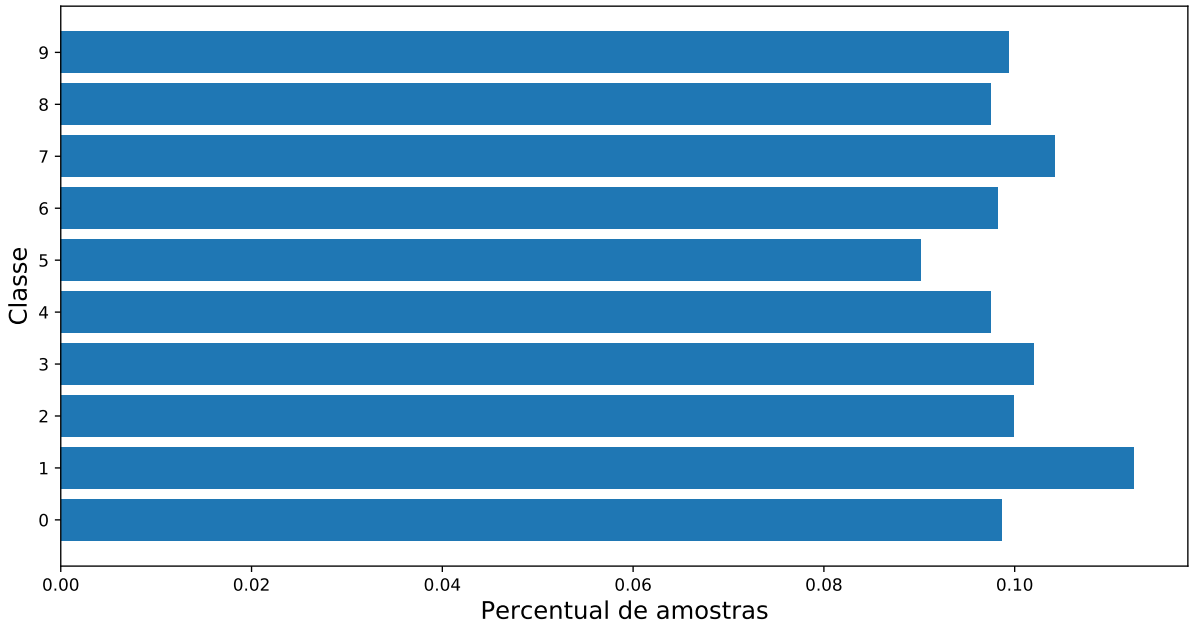

Figura 3 – Apresentação da estratificação dos exemplos dentro do conjunto de dados *MNIST*. A estratificação é feita por classes.

Estas informações são importante, uma vez que delimitarão as abordagens possíveis para métrica de teste, treino e avaliação de generalização de conhecimento dentro desse conjunto de dados – discussão que será apresentada na Seção 3.2.

Por fim, neste projeto o conjunto de dados usado foi obtido através do site oficial e processado utilizando a biblioteca *mnist* disponível na linguagem *Python*.

3 Metodologia

Esta seção apresentará todo o processo metodológico desenvolvido para resolução dos problemas propostos. Serão apresentados os métodos e as arquiteturas escolhidas para abordar cada conjunto de dados. Ademais, serão discutidos os processos de pre-processamento, o motivo para escolha de cada método/arquitetura e a abordagens de treinamento e teste escolhidas. Desta forma, esta seção se divide entre a Seção 3.1, que demonstrará esse desenvolvimento para o conjunto de dados *Íris*, e a Seção 3.2, que apresentará a metodologia desenvolvida para o conjunto de dados *MNIST*.

3.1 Metodologia: Íris

A primeira análise feita foi sobre o problema, visando definir os possíveis métodos a serem utilizados para treinamento e classificação do conjunto de dados. De maneira direta, foi feita a escolha de se implementar duas *Multi-Layer Perceptrons* (MLPs) com arquiteturas distintas. A Seção 3.1.1 discutirá os motivadores para escolhas de cada arquitetura, assim como apresentará em detalhes seus modelos.

Em um segundo momento, frente a escolha da implementação das MLPs, foi realizado um estudo para melhorar o resultado de classificação de ambas MLPs através de um pré-processamento dos dados, buscando ressaltar as características mais relevantes sem enviesar o aprendizado. A Seção 3.1.2 discutirá todo a etapa de pré-processamento realizado para aprimorar os resultados das redes.

Por fim, foi realizado um estudo de possíveis métricas para avaliação das redes, contemplando os pontos levantados na especificação. A Seção 3.1.3 conduzirá toda a discussão acerca do desenvolvimento das métricas.

3.1.1 Arquiteturas das MLPs

A definição das arquiteturas das redes MLPs tomou como base as características do problema, distribuição dos dados e análise da literatura sobre a resolução deste problema (5, 6, 7, 8). Desta forma, buscou-se garantir que estas atingiriam a convergência de maneira rápida sem uma perda de eficiência.

A primeira arquitetura foi definida sobre a premissa de que são necessários, no mínimo, 2 hiper-planos para se separar o conjunto de dados entre as classes disponíveis. Desta forma, seriam necessárias duas camadas escondidas nesta arquitetura, criando, consequentemente, 2 hiper-planos parametrizáveis na rede. Quanto ao número de nós nestas camadas escondidas, optou-se por se utilizar o número de saídas da camada final como re-

ferência. Sendo que o problema possui 3 possíveis classes para classificação, as 2 camadas escondidas desta arquitetura possuirão 3 nós cada. A Figura 4 ilustra essa rede.

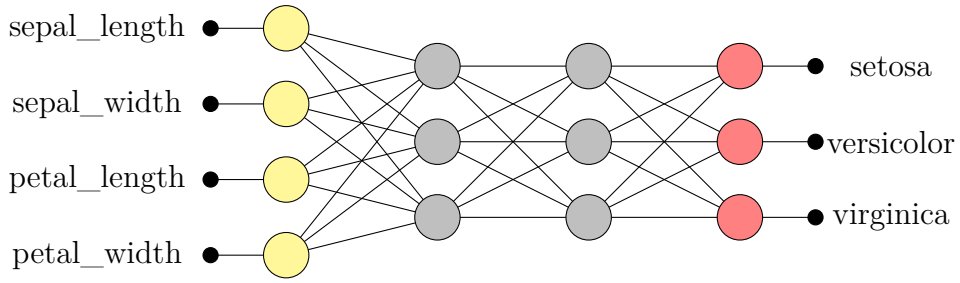

Figura 4 – Ilustração da arquitetura definida para a MLP 1. Os nós em amarelo constituem a camada de entrada, os cinzas representam nós da camada escondida e, os em vermelho, nós da camada de saída. Arquitetura = (4:3:3:3).

Já a segunda arquitetura, foi definida considerando-se a possibilidade de se processar cada dado de entrada sobre uma camada de tamanho igual ao da camada de entrada, isto é, em uma primeira camada escondida de tamanho 4. A adição dessa nova camada escondida possibilita a divisão do espaço de amostras por 3 hiper-planos distintos (um a mais do que o apresentado na arquitetura da MLP 1). Frente a este fato, a definição desta arquitetura é a prospecção de que esta seja capaz de melhor processar os dados de entrada, ponderando-os antes de uma propagação sobre uma camada com o número de nós menor do que o número de entrada e, por tanto, melhor classificar os exemplos disponíveis no conjunto. A Figura 5 ilustra essa rede.

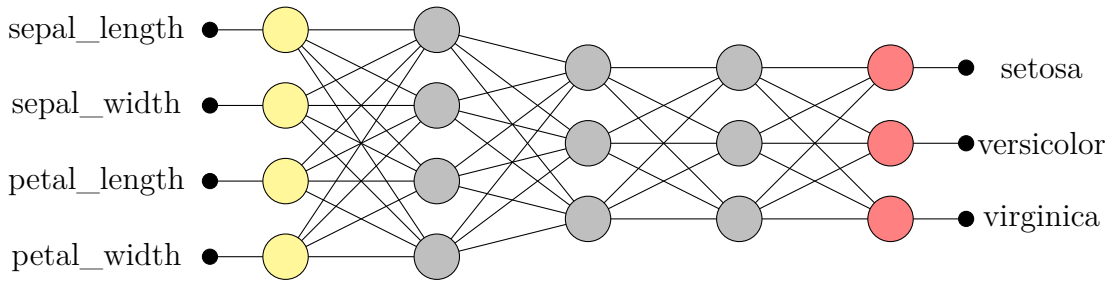

Figura 5 – Ilustração da arquitetura definida para a MLP 2. Os nós em amarelo constituem a camada de entrada, os cinzas representam nós da camada escondida e, os em vermelho, nós da camada de saída. Arquitetura = (4:4:3:3:3).

Elencando, as arquiteturas escolhidas para implementação foram:

- **Arquitetura MLP 1 (4:3:3:3):**
  - 4 neurônios em uma camada de entrada;

  - 2 camadas escondidas com 3 neurônios na primeira camada escondida e 3 na segunda camada, e;
  - 3 neurônios na camada de saída.

- **Arquitetura MLP 2 (4:4:3:3:3):**
  - 4 neurônios em uma camada de entrada;
  - 3 camadas escondidas com 4 neurônios na primeira camada, 3 na segunda camada e 3 na terceira camada, e;
  - 3 neurônios na camada de saída.

3.1.2 Pré-processamento

Com o inicio da fase de análise dos dados para processamento dentro das MLPs, o primeiro passo realizado foi a observação da distribuição dos exemplos disponíveis no conjunto de dados para induzir possíveis abordagens que pudessem melhorar o processo de aprendizado. A Figura 6 apresenta a distribuição dos dados de acordo com seus parâmetros (explorados em pares).

O primeiro ponto levantado para desenvolvimento durante o pre-processamento dos dados, refere-se a escala de distribuição dos dados. Percebeu-se que intervalos de distribuição dos parâmetros eram distintos, fato que, do melhor conhecimento do aluno, pode deteriorar os limites de aprendizado da rede (uma vez que certos pesos desta podem ter ponderamentos em valores muito maiores ou menores que outros devido a falta de normalidade dos valores de entrada, enviesando o conhecimento dentro da rede). Desta forma, decidiu-se realizar uma normalização dos exemplos no conjunto de dados através da técnica de normalização conhecida como "método de normalização via média e desvio padrão" (do inglês, *mean and standard deviation scaling method*).

Pela aplicação deste método, é possível garantir que os intervalos de distribuição dos parâmetros estejam dentro de um intervalo normal (estatístico) ao conjunto de amostras. A Figura 7 apresenta a distribuição dos parâmetros após normalização.

Note que a distribuição dos dados não foi alterada. Contudo, a normalização modificou significativamente os intervalos de distribuição dos parâmetros. Anteriormente, os parâmetros podiam estar distribuídos em um intervalo real $I_a = [0,\ 8]$, com limites aproximados. Após a normalização, o novo intervalo passou a ser definido em um intervalo real $I_b = [-1.6,\ 2]$, com os limites também aproximados. Acredita-se que esta estratégia possibilitará o aprimoramento do aprendizado da rede, aprimorando, consequentemente, a classificação final sem alterações nas arquiteturas das redes. Resultados referentes a esta melhora serão apresentados na Seção 4.1.

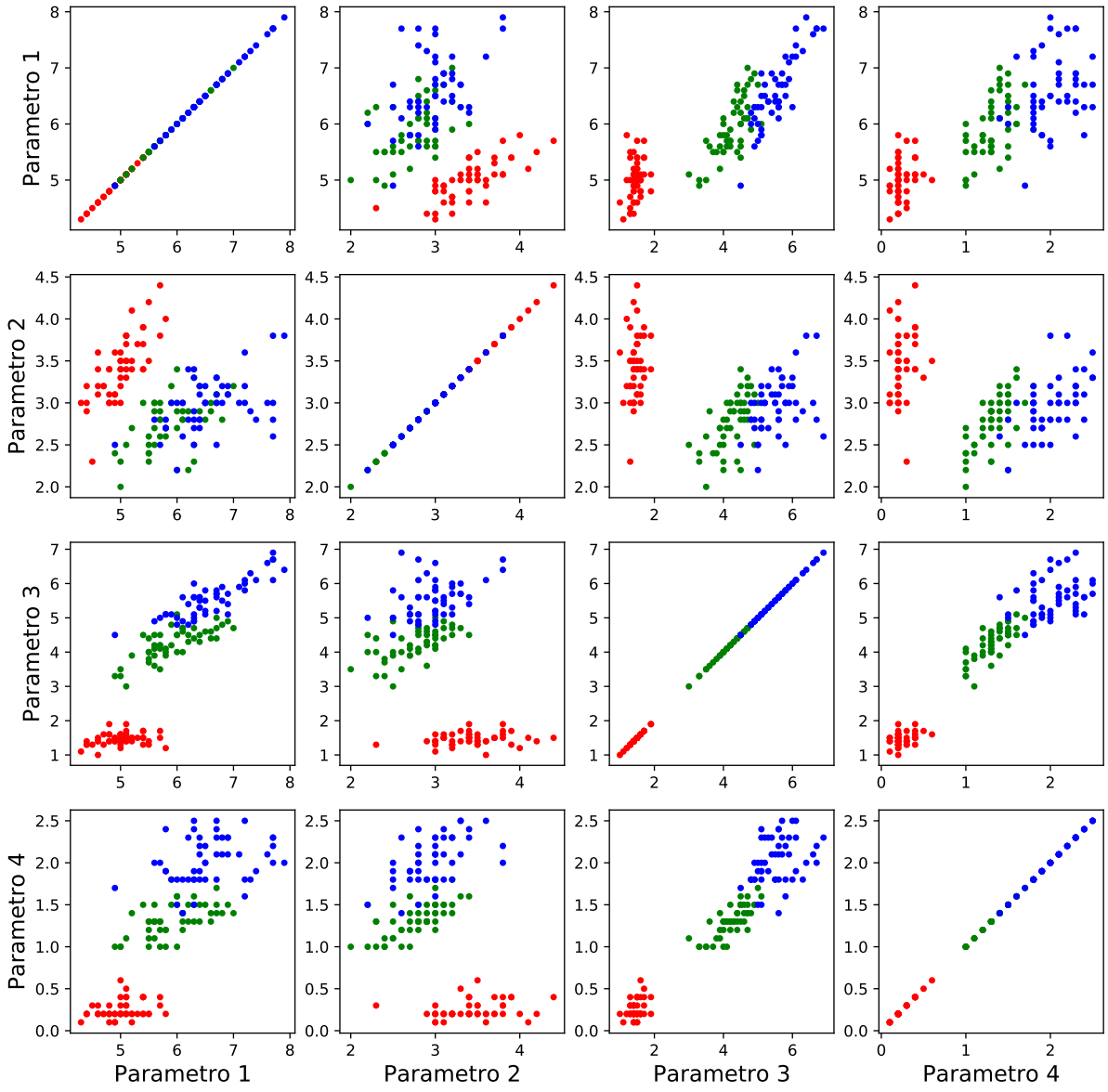

Figura 6 – Distribuição dos dados do conjunto de dados *Íris*. Cada gráfico foi construído utilizando-se um par de parâmetros (para possibilitar a visualização). Os pontos vermelhos representam a classe "Setosa", os verdes a classe "Versicolor" e os azuis a classe "Virgínica".

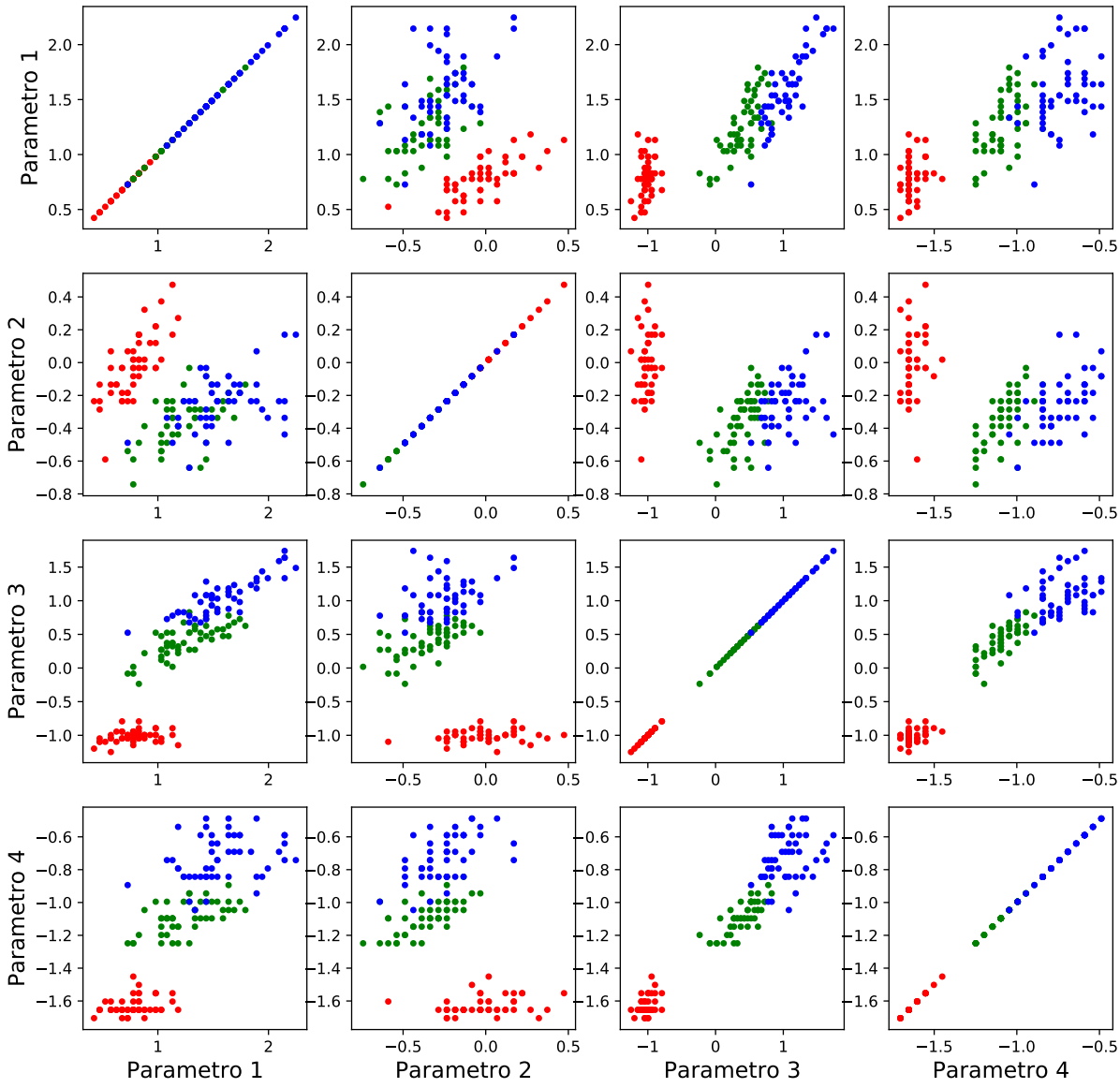

Figura 7 – Distribuição dos dados normalizados do conjunto de dados *Íris*. Cada gráfico foi construído utilizando-se um par de parâmetros (para possibilitar a visualização). Os pontos vermelhos representam a classe "Setosa", os verdes a classe "Versicolor" e os azuis a classe "Virgínica".

3.1.3 Métricas

Por fim, a última etapa da metodologia contempla a definição das métricas para análise e comparação das redes. Será avaliada a métrica de acurácia ("score") média dos modelos através da aplicação do método conhecido como "K-Fold Cross-Validation" Estratificado (do inglês, *Stratified K-Fold Cross-Validation*).

Serão executados 10 processos de K-Fold "cross-validation" Estratificado, com $K = 5$. A escolha desse método de validação se deu, principalmente, pelos componentes disponíveis na implementação da classe utilizada para análise dos modelos MLP e pela característica do conjunto de dados (que é balanceado). O resultado final apresentará a média e o desvio padrão das acurácias de cada modelo durante as 10 execuções de "cross-validation" realizadas.

3.1.4 Implementação

A implementação dos modelos MLP foram feitos através da biblioteca *Sklearn* em *Python* utilizando a classe *MLPClassifier*. O método de validação também está disponível na biblioteca *Sklearn* pela função *cross_validate*. Toda documentação desses métodos podem ser encontrados na página oficial da biblioteca.

Foi utilizada, especificamente, a linguagem *Python* na versão 3 para implementação do código principal e execução dos experimentos.

3.2 Metodologia: MNIST

A primeira análise feita para desenvolvimento dos experimentos relacionados ao conjunto de dados *MNIST* foi sobre o problema que este define, afim de se identificar os possíveis métodos que poderiam ser utilizados para extração de conhecimento da base e classificação do conjunto de dados. Esta análise contou, formalmente, com uma revisão sistemática guiada pelos conhecimentos que o aluno já detém dentro da área. Desta forma, foram selecionadas as implementações de um modelo de Rede Neural Convolucional (*CNN*, do inglês "*Convolutional Neural Network*") e uma Máquina de Vetor de Suporte (*SVM*, do inglês "*Support Vector Machine*") Linear para resolução do problema. Ambos métodos e arquiteturas serão apresentados nas Seções 3.2.1 e 3.2.2, respectivamente.

Ademais, serão apresentadas as metodologias de pré-processamento na Seção 3.2.3 e as métricas escolhidas para validação e avaliação dos modelos na Seção 3.2.4.

Por fim, serão tecidos breves comentários sobre a implementação dos modelos e código principal na Seção 3.2.5.

3.2.1 Rede Neural Convolucional

A escolha da implementação de uma CNN possui uma relação direta ao fato desta representar uma das principais abordagem da literatura para aprendizado e extração de padrões, atingindo, para diversos casos, um grau de generalização satisfatório de alta acurácia (9, 10, 11). Além disso, redes convolucionais apresentam, em sua definição, vantagens para o processamento de matrizes de dados, tal qual uma figura 2-D. O processo de convolução aplicado para aprendizado otimiza o processamento e melhora a extração de padrões relevantes para a classificação desse tipo de dados.

A arquitetura escolhida para implementação da CNN representa um modelo tradicional da literatura com a adição de detalhes que, o aluno, acredita favorecer o aprendizado e generalização dos dados. O modelo tradicional apresenta uma rede com 3 camadas convolucionais com funções logísticas-sigmoid acopladas para ativação e 3 camadas de *Max-Pooling*. Detalhadamente, a rede possui:

- 3 camadas convolucionais:
  - a primeira camada utiliza 32 filtros de dimensão 3x3;
  - a segunda camada utiliza 64 filtros de dimensão 3x3;
  - a terceira camada utiliza 128 filtros de dimensão 3x3;
- 3 camadas para a função de ativação logística-sigmoid;
- 2 camadas *Max-Pooling*:
  - a primeira com *poolings* de tamanho 2x2;
  - a segunda com *poolings* de tamanho 2x2;

A Figura 8 apresenta a ilustração deste modelo utilizado para definir a arquitetura.

O modelo proposto pelo aluno conta com a adição de 4 camadas de *Dropout* na arquitetura, visando evitar problemas de memorização e/ou "*overfitting*" na rede e, consequentemente, favorecer a generalização do aprendizado. As probabilidades de *dropout* utilizadas em cada camada foi de 0.25,0.25,0.4 e 0.3, respectivamente. Ademais, o aluno propõe a utilização da função de ativação de Unidades Lineares Retificadas (*ReLU*, do inglês "*Rectified Linear Units*"). Isto porque, do melhor conhecimento do aluno, a utilização desta função de ativação em redes neurais convolucionais tem se mostrado mais efetiva, uma vez que esta é capaz de lidar com problemas de neurônios "hiper-alimentados". Neurônios "hiper-alimentados" são neurônios que possuem seus pesos de ativação modificados (em alguma etapa do treinamento) para um valor que não mais permitirá sua reativação. Foi utilizado o valor $\alpha = 0.1$ como constante da função de ativação ReLU.

Frente a estas justificativas, a nova arquitetura possui:

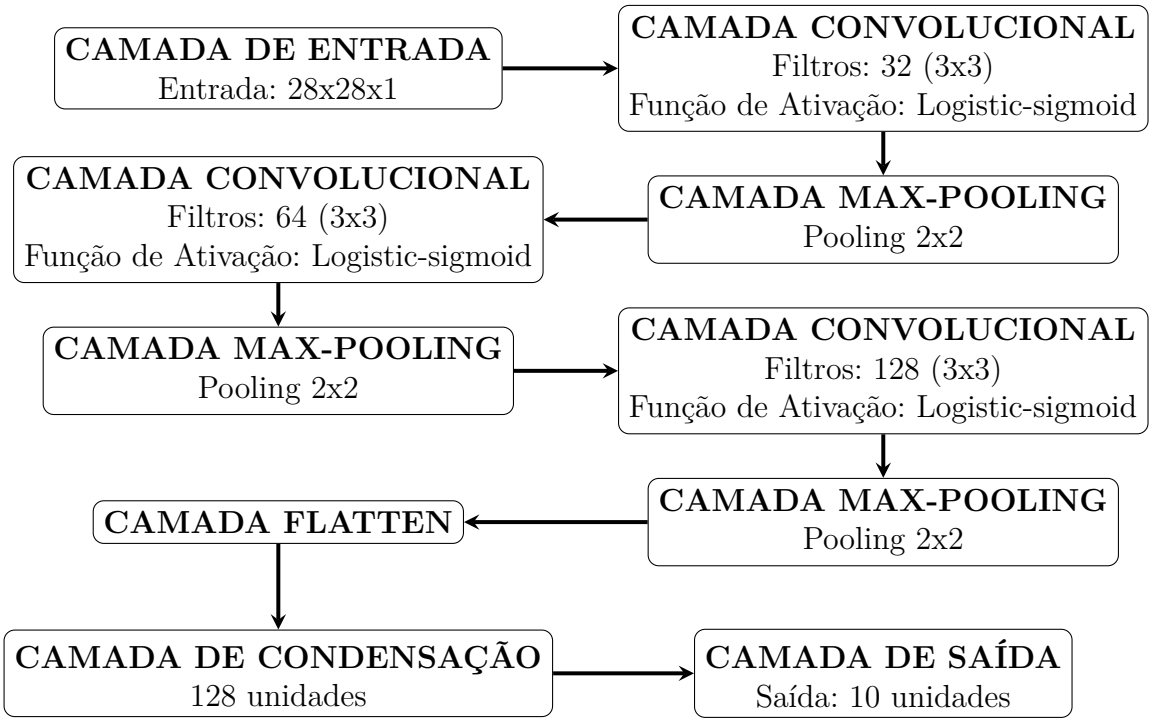

Figura 8 – Ilustração da arquitetura utilizada na implementação da CNN.

NOVA ARQUITETURA PROPOSTA

- 3 camadas convolucionais:
  - a primeira camada utiliza 32 filtros de dimensão 3x3;
  - a segunda camada utiliza 64 filtros de dimensão 3x3;
  - a terceira camada utiliza 128 filtros de dimensão 3x3;
- 3 camadas para a função de ativação ReLU, com $\alpha = 0.1$;
- 2 camadas *Max-Pooling*:
  - a primeira com *poolings* de tamanho 2x2;
  - a segunda com *poolings* de tamanho 2x2;
- 4 camadas *Dropout*:
  - a primeira com probabilidade 0.25 para desativação;
  - a segunda com probabilidade 0.25 para desativação;
  - a terceira com probabilidade 0.4 para desativação;
  - a quarta com probabilidade 0.3 para desativação;

A Figura 9 apresenta esta proposta de aprimoramento da arquitetura tradicional, destacando suas modificações.

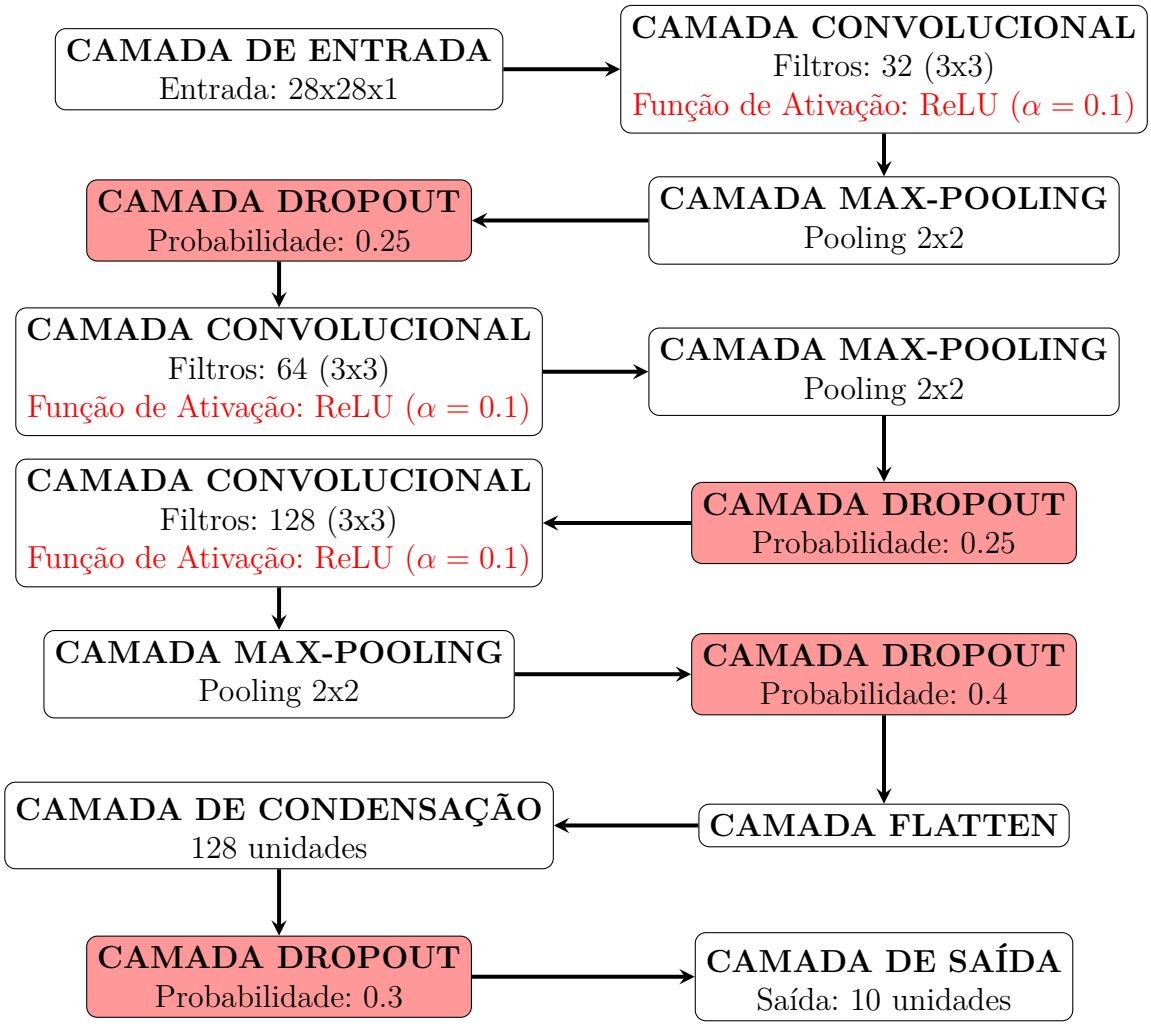

Figura 9 – Ilustração da arquitetura proposta para implementação da CNN.

Assim, foi definido o primeiro modelo a ser utilizado para classificação dos conjuntos de dados, dando inicio a uma nova fase análise visando a definição do segundo candidato requisitado na especificação do projeto.

3.2.2 Máquina de Vetor de Suporte Linear

A decisão de se implementar o método para aprendizado como um modelo SVM pois o ajuste necessário para um funcionamento satisfatório do método como classificador, sobre este conjunto de dados, é mínimo. Assim, pretende-se confrontar os seus resultados com a CNN proposta em quesito de acurácia, tempo e pre-processamento necessário. O modelo SVM-Linear escolhido para implementação possui a seguinte arquitetura:

- Parâmetro de regularização $C = 0.01$;
- Kernel Linear (ordem = 1);
- Comparação para decisão do tipo “Um contra Todos”.

Por fim, reitera-se que é esperado que a SVM alcance uma taxa de generalização e acurácia satisfatória (ainda que menor que do modelo CNN) a um menor custo de modelagem e tempo de treinamento que o modelo convolucional.

Esta escolha contribui para o desenvolvimento de conhecimento e maturidade na área pois, cabe ao profissional responsável, saber analisar as restrições associadas ao seu problema; por diversas vezes, tomando a decisão de escolha de modelo analisando o "trade off" entre tempo investido e acurácia alcançada.

3.2.3 Pré-processamento

O pré-processamento do conjunto de dados *MNIST* contou com 2 fases e foi diferente para cada um dos modelos. Ao analisarmos os dados, tem-se que as imagens já estão centralizadas e normalizadas (quanto à tamanho), como mencionado na descrição do conjunto. Contudo, visando aprimorar o aprendizado, escolheu-se refinar ainda mais estas imagens.

A primeira fase de pré-processamento é responsável por realizar a normalização dos dados quanto à valor de parâmetros. Originalmente, os parâmetros encontram-se distribuídos em um intervalo inteiro $I_c = [0-255]$, como é esperado em imagens no padrão preto e branco ("*Grayscaled*"). Porém, devido à diferença entre o menor e maior valor do intervalo, utilizar esse conjunto de dados sem uma normalização pode causar degeneração do processo de aprendizado, diminuindo a taxa de generalização e, consequentemente, a acurácia do modelo. Assim, foi aplicado o método de normalização via média e desvio padrão. Diretamente, as imagens que antes tinham seus parâmetros dentro do intervalo inteiro I_c, agora os tem dentro de intervalo real que segue uma distribuição normal (de acordo com a média e desvio padrão calculado). Esta abordagem já deve melhorar o aprendizado significativamente (como prospectado para os testes realizados no conjunto de dados *Íris*).

Concluída a normalização, outra ideia que emergiu para aprimoramento do processo de aprendizado foi a redução do espaço parametrizável. Como discutido anteriormente, as imagens do conjunto de dados possuem dimensão 28x28 e, portanto, o dispõe de 784 parâmetros que devem ser estimados para a classificação de uma imagem. Este espaço de parâmetros pode incluir componentes que não são relevantes ou que possuem uma taxa de decisão muito baixa para a realização de uma classificação. Para ilustrar esta ideia, considere que a borda preta em volta dos exemplos (i.e., área fora do centro, de cor padrão preta para todas as imagens) seja dada como entrada parcial para classificação. Nenhuma informação é retirada desses parâmetros (ou pixels) uma vez que eles não apresentam qualquer informação sobre o dígito, sendo possível assim descartá-los no processo de apresendizado de padrões para classificação.

Desta forma, a segunda etapa de pré-processamento do conjunto de dados define a aplicação da técnica de Análise de Componentes Principais (*PCA*, do inglês *Principal Component Analysis*). Basicamente, esta técnica visa extrair as principais componentes que representam um conjunto de dados através de uma análise estatística. A técnica é responsável por apontar os principais parâmetros que caracterizam um conjunto de dados e reduzir este espaço sem grande perda de informação.

Aplicando esta técnica sobre todo o conjunto de dados, é possível se realizar uma análise de como a variância de informação se comporta com a variação do número de parâmetros. A Figura 10 apresenta este comportamento em forma de um gráfico.

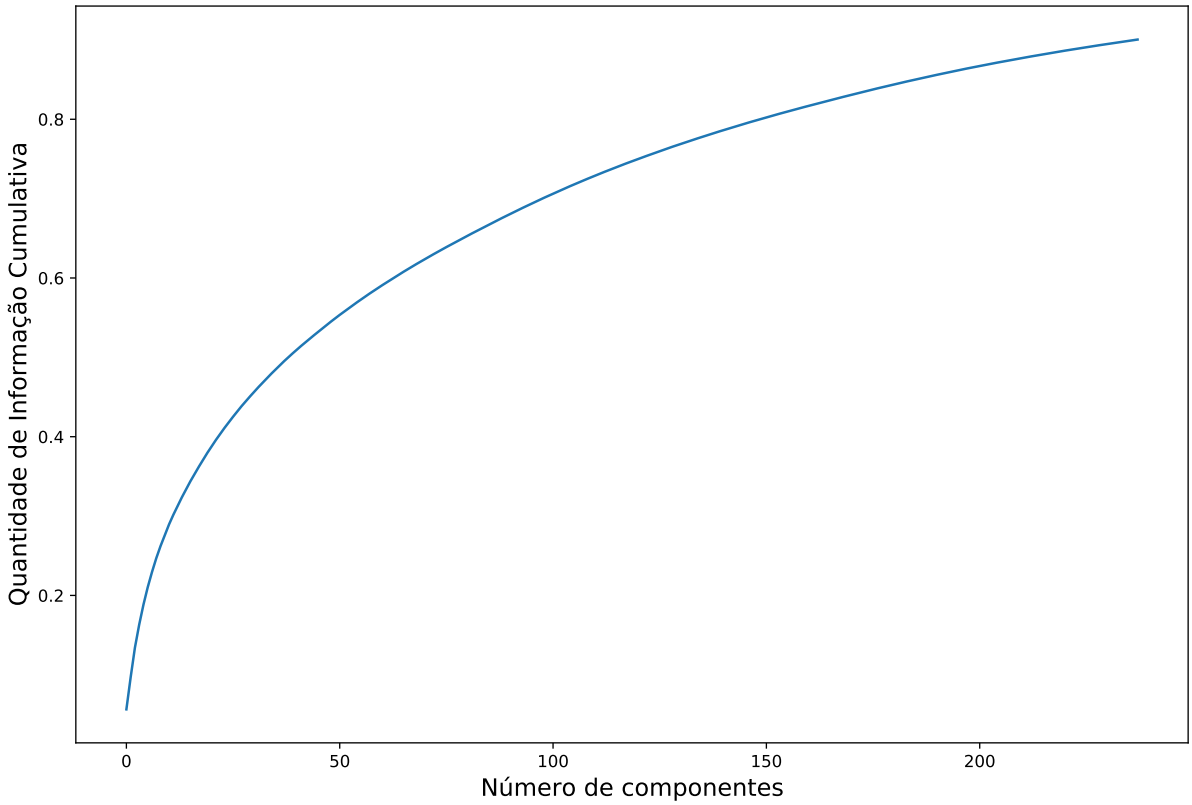

Figura 10 – Gráfico que apresenta a quantidade de informação percentual cumulativa com o aumento de parâmetros sobre um conjunto de dados. Foi aplicado o método PCA para extração destes dados.

Analisando o gráfico, é possível perceber que 90% da informação contida em uma imagem pode ser representada pela utilização de, aproximadamente, 230 parâmetros (ou pixels). Mais precisamente, 90% da informação está contida em 238 pixels, segundo o método PCA. Portanto, é possível se reduzir o espaço paramétrico de 784 em 70% e ainda possuir 90% da informação original representado por, somente, estas componentes principais (30% dos parâmetros). Frente a isso, decidiu-se reduzir o número de parâmetros de 784 pixels para 238. Note que para se representar a imagem utilizando este número de pixels (parâmetros), é necessário a definição de uma área quadrada (sendo 16x16 a mais próxima) e a aplicação de interpolações para reconstruí-la. Contudo, quanto a

treinamento e classificação, esta redução deste espaço proposta deve acelerar o processo de significativamente.

Ademais, é importante destacar que o processamento PCA **não foi utilizado** sobre o conjunto de testes e treinamento para o modelo CNN. Foi tomada esta decisão pois a perda de informação pode afetar significativamente o processo de convolução e interpolação aplicado pelas redes convolucionais, bastando somente a normalização de seus conjuntos para garantir a eficiência do método. Assim, concluiu-se o processo de pre-processamento da base de dados.

3.2.4 Métricas

Por fim, a última etapa da metodologia pretende definir as métricas para análise e comparação dos modelos escolhidos. Serão avaliadas 5 métricas distintas: acurácia, perda, precisão, *recall* e a taxa de aprendizado do modelo. Todos os resultados representarão o valor médio extraído da aplicação do método de validação conhecido como "K-Fold Cross-Validation" Estratificado (do inglês, *Stratified K-Fold Cross-Validation*), com $K = 5$.

Ademais, diferentemente do processo adotado para validação do conjunto de dados *Íris*, será executada somente uma rodada de aplicação do método (uma vez que o treinamento pelo conjunto de dados *MNIST* tende a demorar significativamente mais para executar, inviabilizando novas rodadas).

3.2.5 Implementação

A implementação do modelo CNN foi feita através da biblioteca *Tensorflow* pelo modulo *Keras*, o qual disponibiliza a classe *Sequential*. Esta classe permite a adição de camadas sequencialmente para construção da rede, fato essencial para o desenvolvimento deste projeto. Para implementação da SVM, foi utilizada a classe *SVC*, disponível no pacote do *Sklearn* no módulo *svm*. Os métodos de validação aplicados foram importados da biblioteca *Sklearn* em seu módulo *metrics*. Todos os métodos e documentações podem ser encontrados na página oficial da biblioteca.

Foi utilizada, especificamente, a linguagem *Python* na versão 3 para implementação do código principal e execução dos experimentos.

4 Resultados

Esta seção visa apresentar os dados colhidos para validação dos métodos propostos e para avaliação dos resultados obtidos nas classificações. Para avaliação, também será conduzida uma comparação entre os resultados obtidos e os resultados presentes nas referências. Desta forma, a Seção 4.1 visa apresentar a validação e resultados para o conjunto de dados *Íris* e a Seção 4.2 para o conjunto de dados *MNIST*.

4.1 Resultados: Íris

Primeiramente, foi conduzido a validação dos métodos através do K-Fold Estratificado Cross-Validation com o conjunto de dados não normalizado. Foi utilizado o parâmetro $K = 5$ e foram executados 10 processos de validações K-Fold, calculando-se assim a acurácia média e a variância resultantes de todos os processos. O resultado é apresentado na Tabela 1.

	Teste		Treino	
	Média	Desvio Padrão	Média	Desvio Padrão
MLP 1	0.883	± 0.052	0.923	± 0.051
MLP 2	0.907	± 0.036	0.9153	± 0.068

Tabela 1 – Apresentação das acurácias médias obtidas por cada classificador após 10 repetições do processo de validação K-Fold Estratificado sobre o conjunto de dados não-normalizado *Íris*.

Analisando a tabela, é possível notar que a MLP 2 obteve um melhor resultado médio na classificação de amostras nunca vistas do que a MLP 1 (considerando 50 conjuntos de teste-treino distintos). Ademais, dado a abordagem pela validação K-Fold e observando os resultados obtidos, não há qualquer sinal de "over" ou "underfitting" neste processo.

Em um segundo momento, foi conduzida a mesma estratégia de validação. Contudo, agora foi utilizado o conjunto de dados normalizados segundo a estratégia apresentada na Seção 3.1. O resultado é apresentado na Tabela 2.

	Teste		Treino	
	Média	Desvio Padrão	Média	Desvio Padrão
MLP 1	0.955	± 0.014	0.988	± 0.011
MLP 2	0.919	± 0.067	0.943	± 0.070

Tabela 2 – Apresentação das acurácias médias obtidas por cada classificador após 10 repetições do processo de validação K-Fold Estratificado sobre o conjunto de dados normalizado *Íris*.

Note que, com a normalização dos dados, os resultados de classificação no teste e treino foram aprimorados e, agora, o modelo MLP 1 passou a ter um melhor resultado geral para classificação do que o modelo MLP 2. Ademais, nenhum indício de "under" ou "overfitting" aparecem nestes resultados.

Assim, tem-se que, com a normalização de dados, o melhor resultado a ser obtido é através da aplicação do modelo MLP 1; apresentando uma taxa de generalização satisfatória (com uma acurácia média de 95% de acerto nas classificações). Corroborando à ideia de que esta taxa de generalização é satisfatória, do melhor conhecimento do aluno, os melhores modelos MLP presentes na literatura alcançam uma acurácia próxima à 97% (5, 12) e outros modelos, como o "K-Nearest Neighbors", conseguem alcançar acurácia próxima à 96 à 98% (13, 14)

Por fim, uma matriz de confusão foi construída para apresentar os resultados da melhor rede obtida. A Figura 11 apresenta esta matriz.

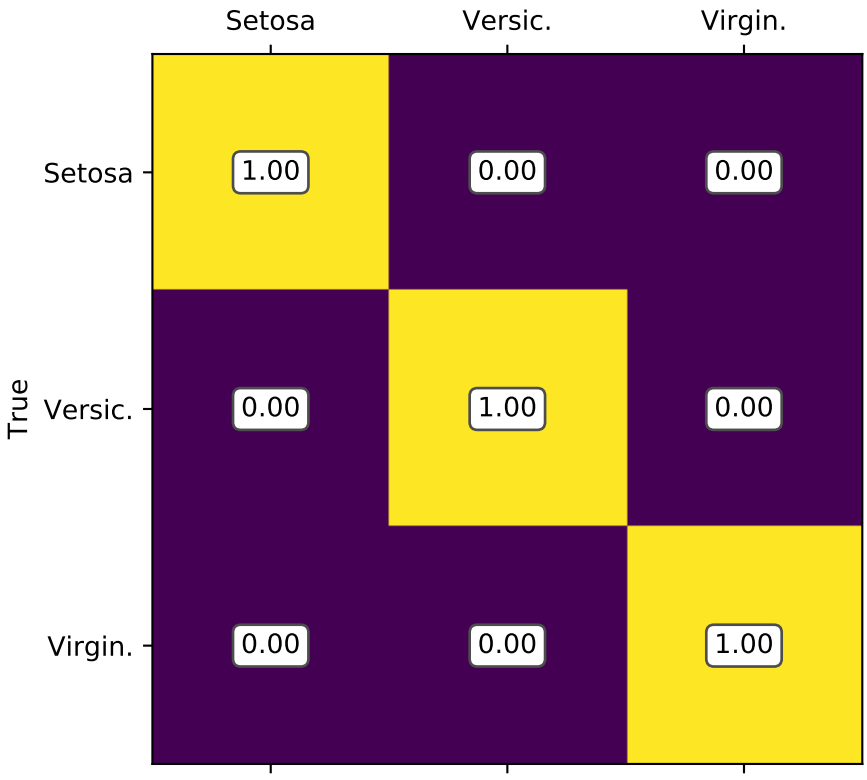

Figura 11 – Matriz de confusão obtida com a classificação feita pelo melhor modelo encontrado durante o desenvolvimento e análise do projeto.

Ressalva-se que o fato do melhor classificador ter conseguido alcançar uma acurácia de 100% não define que o modelo tenha essa generalização. Contudo, apresenta-se este a fim de apontar que, para alguns casos, o classificador possui performance notável e satisfatória, conforme esperado dado grau de generalização alcançado.

4.2 Resultados: MNIST

Primeiramente, foram coletados os resultados referentes a aplicação da CNN sobre o conjunto de dados. Considerando a divisão do conjunto de dados na proporção 15%/85% para teste e treinamento, respectivamente, e um período de 20 épocas para treinamento, o primeiro resultado colhido foi a taxa de aprendizado do modelo. A Figura 12 apresenta as curvas de aprendizado associadas ao modelo.

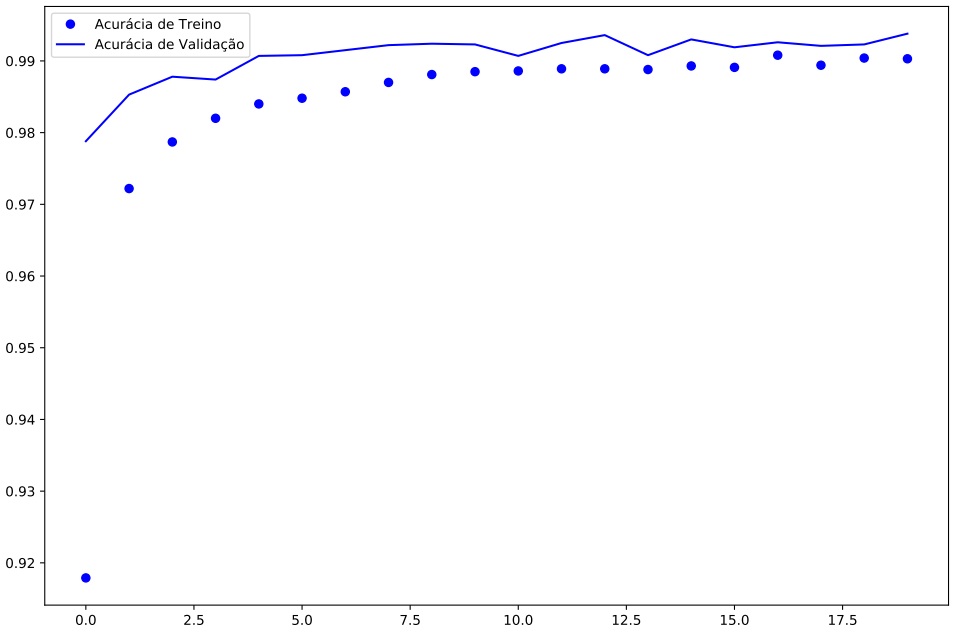

Figura 12 – Apresentação das curvas de acurácia do modelo CNN.

Observando as curvas, é possível notar que, após a 13 época, a curva de acurácia do modelo tende a estabilizar, tanto para o conjunto de treino, quanto para o conjunto de validação. Ademais, é possível argumentar que não há sinais de "under" ou "overfitting" uma vez que a acurácia para treino e teste estão próximas para todas as época, exceto a primeira. Em outras palavras, é possível notar uma generalização do conhecimento obtido, sem a memorização dos exemplos.

Após esta analise visual do progresso de aprendizado da rede, foram coletados sobre as métricas de acurácia, perda, precisão e *recall* da rede. A Tabela 3 apresenta a acurácia e perda média sobre o conjunto de testes e a Tabela 4 apresenta as métricas de precisão, *recall* e o suporte da rede para cada classe.

Acurácia	**Perda**
0.9938095	0.023752805635468187

Tabela 3 – Acurácia e Perda média da CNN sobre o conjunto de teste.

	Precisão	**Recall**	**Suporte**
Classe 0	0.99	1.00	1020
Classe 1	1.00	1.00	1242
Classe 2	1.00	0.99	1079
Classe 3	0.99	0.99	1081
Classe 4	0.99	0.99	983
Classe 5	0.99	0.99	982
Classe 6	0.99	0.99	983
Classe 7	0.99	1.00	1100
Classe 8	0.99	0.99	993
Classe 9	0.99	0.99	1037

Tabela 4 – Precisão, *Recall* e Suporte da CNN sobre o conjunto de teste.

Em um segundo momento, foram coletados os resultados referentes a aplicação da SVM-Linear sobre o conjunto de dados. Considerando a divisão do conjunto de dados na proporção 15%/85% para teste e treinamento, respectivamente, o primeiro resultado colhido para este modelo foi sua taxa de aprendizado. A Figura 13 apresenta as curvas de aprendizado associadas ao modelo.

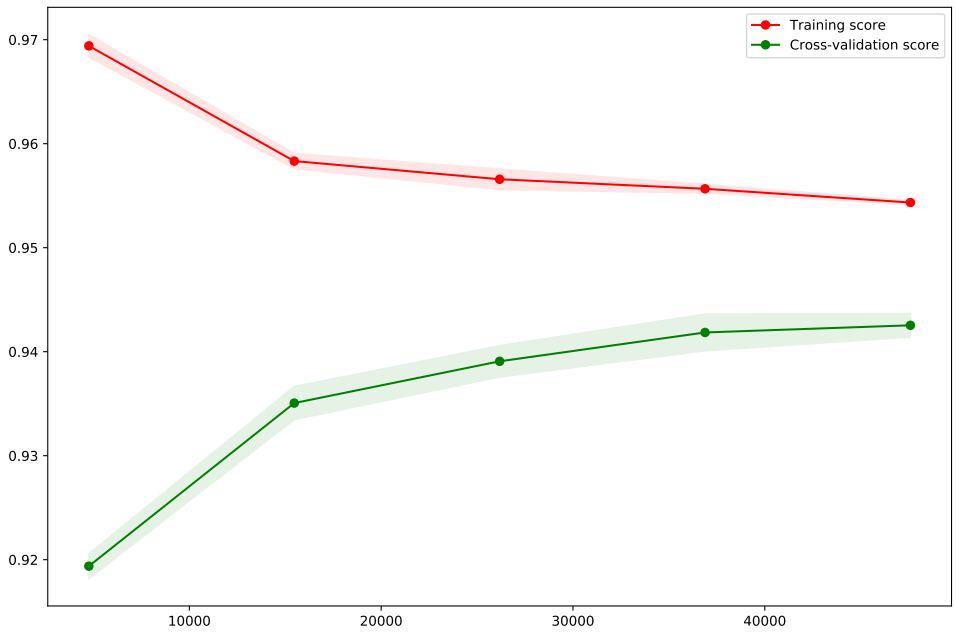

Figura 13 – Apresentação das curvas de acurácia do modelo SVM-Linear com o aumento do número de exemplos utilizados para treino.

Realizando uma análise visual das curvas apresentadas no gráfico, não se nota qualquer indício de "over" ou "underfitting" do modelo. Ambas as curvas (de acurácia no treinamento e pelo "*cross-validation*") encontram-se próximas e convergindo a um ponto de equilíbrio, indicando baixo índice de memorização dos padrões e boa capacidade de generalização do conhecimento. Ademais, o modelo mostra-se capaz de aprimorar os resultados de classificação na etapa de teste conforme mais exemplos são amostrados na etapa

de treinamento, corroborando para definir que o desempenho do modelo é satisfatório na resolução do problema proposto.

Após esta analise, foi desenvolvida a mesma análise feita com o modelo CNN pela coleta das métricas de acurácia, perda, precisão e *recall* do modelo SVM-Linear. A Tabela 5 apresenta a acurácia e perda média sobre o conjunto de testes e a Tabela 6 apresenta as métricas de precisão, *recall* e o suporte da rede para cada classe.

Acurácia	**Perda**
0.9420952380952381	0.0579047619047619

Tabela 5 – Acurácia e Perda média da SVM-Linear sobre o conjunto de teste.

	Precisão	**Recall**	**Suporte**
Classe 0	0.96	0.98	1020
Classe 1	0.97	0.99	1242
Classe 2	0.93	0.94	1079
Classe 3	0.92	0.92	1081
Classe 4	0.93	0.94	983
Classe 5	0.94	0.91	982
Classe 6	0.96	0.96	983
Classe 7	0.94	0.95	1100
Classe 8	0.93	0.90	993
Classe 9	0.93	0.92	1037

Tabela 6 – Precisão, *Recall* e Suporte da SVM sobre o conjunto de teste.

Frente a estes resultados individuais, iniciou-se o estudo sobre a generalização do conhecimento e eficiência do método. Para isto, considerou-se como principal base avaliativa a comparação dos resultados alcançados com os principais resultados catalogados no site oficial do conjunto MNIST. Para realizar esta comparação, os métodos foram enumerados e ordenados do menor erro ao maior erro catalogado. A Tabela 7 apresenta alguns destes resultados e métodos aplicados sobre o conjunto de dados *MNIST*, visando facilitar a comparação.

Como é possível notar na tabela, a CNN proposta neste documento se equiparou a taxa de erro associada ao 17^o melhor método catalogado, superando todas as outras propostas disponíveis no site oficial *MNIST*; apresentando uma taxa de erro de, aproximadamente, 0.62%. Por outro lado, o modelo SVM-Linear superou somente 4 dos exemplos disponíveis na página, apresentando um erro de 5.79% sobre o conjunto de teste.

Contudo, o tempo necessário para treinamento e classificação do conjunto de dados utilizando a SVM-Linear foi significativamente menor do que para o modelo CNN, a um custo de perda de acurácia. A Tabela 8 apresenta o tempo de execução necessário dos métodos para finalizar o seu treinamento.

Posição da Classificação	**Método**	**Pré-processamento**	**Taxa de Erro Teste (%)**
1	committee of 35 conv. net, 1-20-P-40-P-150-10 [elastic distortions]	width normalization	0.23
5	large conv. net, unsup. pretraining [elastic distortions]	none	0.39
10	K-NN with non-linear deformation (IDM)	shiftable edges	0.54
17	large conv. net, unsup features [no distortions]	shiftable edges none	0.62
17-18	CNN	scaling	0.62
18	K-NN, shape context matching deformation (IDM)	shape context feature extraction	0.63
20	Virtual SVM, deg-9 poly, 1-pixel	jittered, deskewing	0.68
25	deep convex net, unsup pre-training [no distortions]	none	0.83
50	K-NN, L3	deskewing, noise removal, blurring	1.73
65	K-NN, Euclidean (L2)	none	5.79
65-66	SVM-Linear	scaling, PCA	5.79
66	pairwise linear classifier	deskewing	7.6

Tabela 7 – Erro percentual de classificação associado a cada método na classificação do conjunto de dados *MNIST*. Os métodos estão organizado em ordem crescente de erro. As entradas em vermelhas destacam as arquiteturas propostas neste documento e suas posições são definidas comparativamente.

Método	**Tempo de Execução (s)**	**Tempo de Execução (min)**
CNN (Dropout)	3264.302	54min 24s
SVM-Linear	966.506	16min 6s

Tabela 8 – Tempo de execução necessário para treinamento de cada modelo proposto.

Note que a SVM-Linear consegue alcançar uma taxa de erro de 5.79% na classificação do conjunto de dados com um processo de treinamento de 16 minutos e 6 segundos, enquanto o modelo CNN necessita de 54 minutos e 24 segundos para alcançar uma taxa de erro de 0.62%. Relativamente, tem-se que a SVM-Linear possui um erro, aproximadamente, 9.3 vezes maior do que o modelo CNN, porém necessita de um tempo 3 vezes menor para treinamento. Consequentemente, em um cenário onde existem restrições de tempo para começar o processo de classificação, o modelo SVM pode ser suficiente e satisfatório.

Por fim, acredita-se que ambos métodos alcançaram um grau de generalização satisfatório sobre o conjunto de dados, apresentando taxas de acurácias suficientemente altas e nenhum sinal de memorização de padrões (o que valida as modelagens, possibilitando futuros trabalhos para aprimoramento destes). Este fato, demonstra a importância de um estudo sobre o problema, assim como da realização de uma boa modelagem que pode levar a resultados ótimos definidos sobre as restrições impostas (que podem ser maior taxa de acerto ou menor tempo, dependendo do cenário).

5 Conclusão

Redes Neurais e técnicas de aprendizado profundo representam uma área de conhecimento e aplicação de extrema importância para a comunidade científica e profissionais na área da computação. Através da correta modelagem, escolha de paradigma e definição de arquitetura, é possível se resolver problemas complexos de maneira satisfatória, trazendo respostas eficientes e que respeitam restrições impostas no contexto de atuação.

Este documento preocupou-se em apresentar estes desafios e desenvolver a metodologia necessária para solucionar problemas tradicionais da literatura relacionadas ao contexto. Pela análise e classificação dos conjuntos de dados *Íris* e *MNIST*, foram implementados 4 modelos de aprendizado capazes de solucionar estes problemas. Os problemas passaram por uma profunda análise para modelagem e uma fase de pré-processamento dos dados antes da aplicação dos métodos implementados.

Foram implementadas 2 modelos Multi-Layer Perceptron com arquiteturas distintas para resolução da classificação *Íris*. A partir de um pré-processamento dos dados, foi possível se alcançar uma taxa de acerto de 95.5% sobre o conjunto de dados utilizando a arquitetura da MLP 1 definida. Esta, por sua vez, sobrepôs o resultado apresentado pela MLP 2 de 91.9% que, comparativamente, representa uma MLP com mais camadas e neurônios para aprendizado.

Para classificação do conjunto de dados *MNIST*, foram implementados um modelo de Rede Neural Convolucional e um modelo *Support Vector Machine-Linear*. Contando com uma normalização sobre os parâmetros das imagens, a arquitetura CNN proposta alcançou uma taxa de acerto de 99.4% sobre o conjunto de dados. Já o modelo SVM-Linear apresentou uma taxa de acerto de 94.2% com a vantagem de um processo de treinamento 3 vezes mais veloz do que o apresentado pela CNN.

Desta forma, durante o projeto foi possível analisar e desenvolver diferentes metodologias para validação e avaliação de redes neurais aplicadas no contexto de classificações para problemas com treinamento supervisionado. Acredita-se que este primeiro projeto, corroborou para aprimoramento, desenvolvimento e maturação da teoria quanto a conhecimento aplicado, contemplando todos os assuntos apresentados até o presente momento na disciplina.

Referências

1 CHEN, D. Research on traffic flow prediction in the big data environment based on the improved rbf neural network. *IEEE Transactions on Industrial Informatics*, IEEE, v. 13, n. 4, p. 2000–2008, 2017. Citado na página 5.

2 ANGELOV, P. P.; GU, X. Deep rule-based classifier with human-level performance and characteristics. *Information Sciences*, Elsevier, v. 463, p. 196–213, 2018. Citado na página 5.

3 ANDERSEN, P.-A.; GOODWIN, M.; GRANMO, O.-C. Deep rts: a game environment for deep reinforcement learning in real-time strategy games. In: IEEE. *2018 IEEE conference on computational intelligence and games (CIG)*. [S.l.], 2018. p. 1–8. Citado na página 5.

4 LI, C. et al. Deep speaker: an end-to-end neural speaker embedding system. *arXiv preprint arXiv:1705.02304*, 2017. Citado na página 5.

5 PATEL, K.; VALA, J.; PANDYA, J. Comparison of various classification algorithms on iris datasets using weka. *Int. J. Adv. Eng. Res. Dev.(IJAERD)*, v. 1, n. 1, 2014. Citado 2 vezes nas páginas 10 e 23.

6 SWAIN, M. et al. An approach for iris plant classification using neural network. *International Journal on Soft Computing*, Academy & Industry Research Collaboration Center (AIRCC), v. 3, n. 1, p. 79, 2012. Citado na página 10.

7 ARYA, V.; RATHY, R. An efficient neuro-fuzzy approach for classification of iris dataset. In: IEEE. *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)*. [S.l.], 2014. p. 161–165. Citado na página 10.

8 PÉREZ, N. F.; FERRÉ, J.; BOQUÉ, R. Calculation of the reliability of classification in discriminant partial least-squares binary classification. *Chemometrics and Intelligent Laboratory Systems*, Elsevier, v. 95, n. 2, p. 122–128, 2009. Citado na página 10.

9 LI, G. et al. Hand gesture recognition based on convolution neural network. *Cluster Computing*, Springer, v. 22, n. 2, p. 2719–2729, 2019. Citado na página 16.

10 XIN, M.; WANG, Y. Research on image classification model based on deep convolution neural network. *EURASIP Journal on Image and Video Processing*, Springer, v. 2019, n. 1, p. 40, 2019. Citado na página 16.

11 BEN-ARI, R.; KISILEV, P.; SULAM, J. *Classifying medical images using deep convolution neural network (cnn) architecture*. [S.l.]: Google Patents, 2019. US Patent App. 15/697,454. Citado na página 16.

12 PLOJ, B.; HARB, R.; ZORMAN, M. Border pairs method—constructive mlp learning classification algorithm. *Neurocomputing*, Elsevier, v. 126, p. 180–187, 2014. Citado na página 23.

13 YIGIT, H. A weighting approach for knn classifier. In: IEEE. *2013 international conference on electronics, computer and computation (ICECCO)*. [S.l.], 2013. p. 228–231. Citado na página 23.

14 ACI, M.; İNAN, C.; AVCI, M. A hybrid classification method of k nearest neighbor, bayesian methods and genetic algorithm. *Expert Systems with Applications*, Elsevier, v. 37, n. 7, p. 5061–5067, 2010. Citado na página 23.