UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
ELE0606 - TÓPICOS ESPECIAIS EM INTELIGÊNCIA ARTIFICIAL

Docente:
José Alfredo Ferreira Costa

Autores:
Ayslenon Câmara de Moura Silveira
Vinícius Venceslau Venancio da Penha

Redes Neurais MLPs

Natal - RN
19 de dezembro de 2023

RESUMO

O relatório abordou tópicos essenciais, incluindo Redes Multilayer Perceptron (MLPs) e sua arquitetura, o Neurônio Artificial e seu papel nas redes neurais, exemplificando a aplicação desta temática por meio da implementação prática em Python com a base de dados **Yale Face Database**.

Em suma, enfatizou-se a importância da compreensão desses conceitos para aplicações práticas e análise de dados.

Sumário

Lista de Figuras

1 Introdução

As **Redes Multilayer Perceptron (MLPs)** representam uma categoria fundamental de redes neurais artificiais, amplamente reconhecidas por sua aplicabilidade em tarefas de aprendizado supervisionado, tais como classificação e regressão.

Este relatório focaliza a aplicação das MLPs na análise da renomada base de dados **Yale Face Database**. Esta base detém uma variedade de expressões faciais humanas, além de características específicas presentes nos retratos fotográficos.

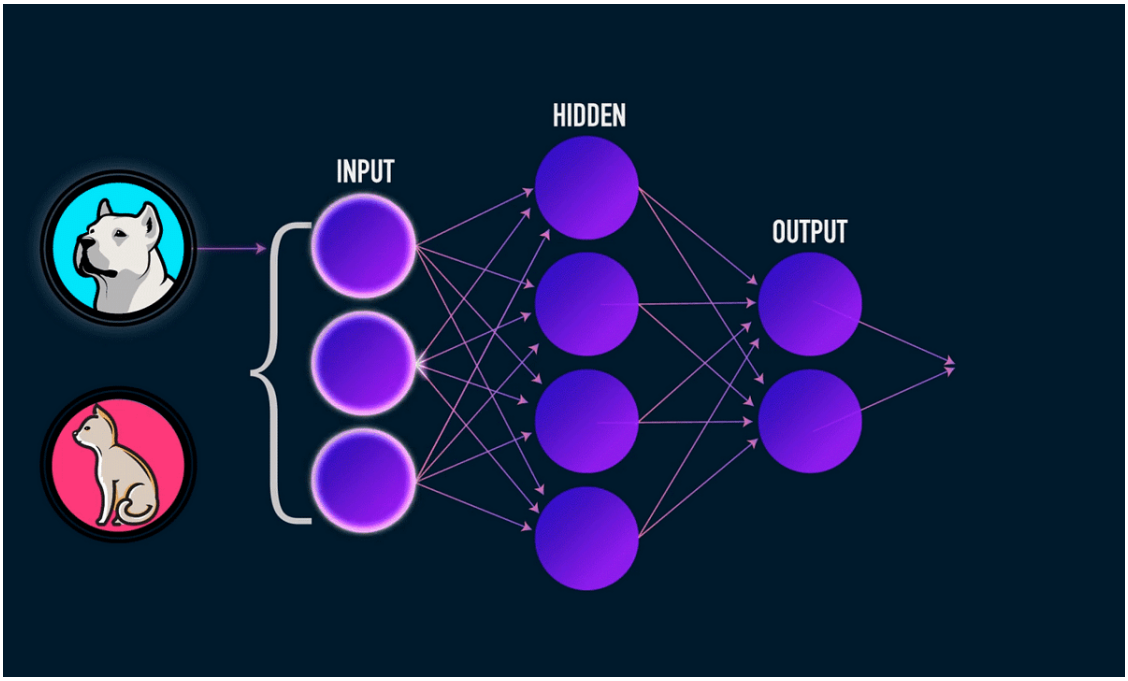


Figura 1: Estrutura de rede neural feedforward.

Nesse contexto, a estrutura básica de uma rede neural **feedforward**, ilustrada na Figura 1, reflete o princípio fundamental das MLPs. O termo "feedforward"refere-se a redes neurais em que as informações fluem em uma direção única, da camada de entrada à camada de saída, sem ciclos ou realimentação.

A rede MLP (Multilayer Perceptron) é um exemplo específico dessas redes feedforward, caracterizada por múltiplas camadas de neurônios (ou camadas ocultas) situadas entre a camada de entrada e a camada de saída. Esta configuração em camadas permite uma modelagem mais sofisticada de relações complexas nos dados, tornando-a especialmente adequada para análises detalhadas e extrapolação de padrões em conjuntos de dados complexos, como a Yale Face Database.

Através deste estudo, busca-se explorar o potencial das MLPs na análise de expressões faciais e características específicas presentes nesta base de dados, visando não apenas a classificação, mas também a compreensão mais profunda das nuances presentes nas imagens.

2 Desenvolvimento

2.1 O Neurônio Artificial

Uma rede neural é composta por neurônios artificiais, unidades fundamentais de processamento. Cada neurônio recebe um conjunto ponderado de entradas, realiza uma transformação linear dessas entradas e aplica uma função de ativação para produzir a saída.

O funcionamento de um neurônio artificial pode ser descrito da seguinte forma:

1. **Entradas:** Cada neurônio recebe um vetor de entradas, denotado, por exemplo, por:
$$X = [x_1, x_2, ..., x_n]$$
onde n é o número de entradas no sistema.

2. **Pesos:** Cada entrada é associada a um peso correspondente, representado como:
$$W = [w_1, w_2, ..., w_n]$$
É importante notar que, neste caso, n ainda se refere ao número de entradas no sistema. Os pesos indicam a importância relativa das entradas para o neurônio.

3. **Operação Linear:** O neurônio realiza uma soma ponderada das entradas multiplicadas pelos pesos, expressa matematicamente como:
$$Z = \sum_{i=1}^{n} (x_i \cdot w_i)$$

4. **Função de Ativação:** Após a operação linear, o neurônio aplica uma função de ativação $f(Z)$ ao resultado Z para produzir a saída Y.

A função de ativação introduz não linearidade à operação do neurônio, sendo exemplos comuns a função sigmóide, ReLU (Rectified Linear Unit) e tangente hiperbólica (tanh).

2.2 Arquitetura das Redes MLP

As Redes Multilayer Perceptron (MLPs) consistem em várias camadas de neurônios, abrangendo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída, todas densamente interconectadas.

2.2.1 Camada de Entrada

Na camada de entrada, a quantidade de neurônios pode corresponder ao número de expressões faciais ou características presentes na imagem sob análise. Essa camada inicial desempenha o papel crucial de ser o ponto de entrada das informações na rede neural.

Para exemplificar esse conceito, em uma situação de classificação de dígitos manuscritos, os nós na camada de entrada podem ser considerados como codificadores da intensidade dos pixels em áreas específicas da imagem. Essa codificação da intensidade dos pixels permite que a rede capture informações detalhadas e essenciais para realizar a classificação de maneira eficaz.

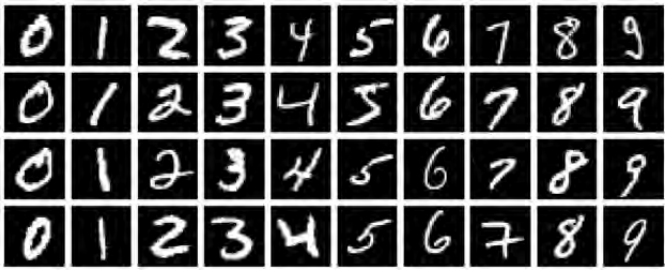


Figura 2: Amostras de imagens do banco de dados de dígitos manuscritos MNIST.

2.2.2 Camadas Ocultas

As camadas intermediárias, localizadas entre a camada de entrada e a camada de saída, são comumente designadas como "intermediárias" devido à sua falta de conexão direta com as entradas ou saídas primárias do sistema. Cada uma dessas camadas é composta por uma quantidade variável de unidades, também conhecidas como neurônios.

Essas unidades processam os dados provenientes das camadas anteriores, aplicando transformações não lineares e transmitindo os resultados à camada subsequente. É a combinação e interação entre múltiplas camadas intermediárias que confere às MLPs a capacidade singular de aprender representações cada vez mais complexas e abstratas dos dados originais.

2.2.3 Camada de Saída

A camada de saída é a última etapa da rede neural e sua configuração varia de acordo com a natureza do problema a ser solucionado. Em tarefas de classificação binária, por exemplo, é comum ter uma única unidade de saída, a qual representa a probabilidade da classe positiva. Em problemas de classificação multiclasse, por outro lado, cada classe possível é associada a uma unidade específica na camada de saída.

Essa camada final é responsável por gerar previsões ou resultados finais, baseando-se nas informações processadas pelas camadas intermediárias. Essencialmente, é a partir desses cálculos finais que a rede neural emite suas conclusões ou predições.

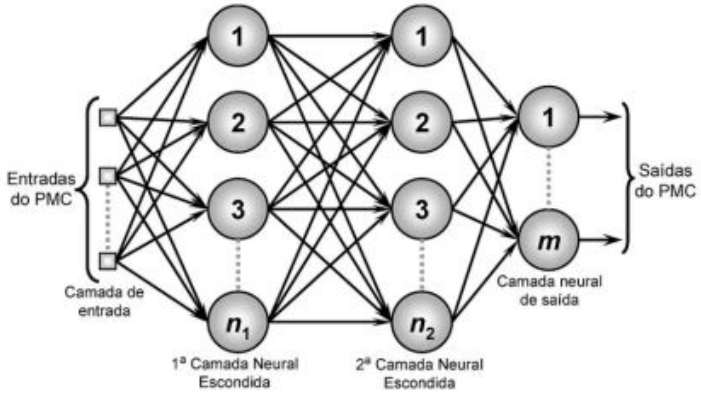


Figura 3: Redes Perceptron de Múltiplas Camadas (PMC).

2.3 Algoritmo em Python

Preparação do Ambiente e Carregamento dos Dados:

Nesta etapa do código, é executada a leitura e o carregamento do arquivo no formato .zip, que contém a base de dados a ser analisada no projeto.

```
import zipfile
import os

# Caminho para o arquivo ZIP upado no Google Colab
caminho_arquivo_zip = '/content/Yale_Face_Database.zip'

# Caminho para o diretorio de destino que ira conter os arquivos
    extraidos
caminho_destino = './Yale_Face_Extraido'

# Certifique-se de que o diretorio de destino exista ou crie-o
os.makedirs(caminho_destino, exist_ok=True)

# Extrair o arquivo ZIP
with zipfile.ZipFile(caminho_arquivo_zip, 'r') as zip_ref:
    zip_ref.extractall(caminho_destino)

print(f'Arquivo ZIP extraido em: {caminho_destino}')
```

Listing 1: Importação dos dados.

```
Arquivo ZIP extraido em: ./Yale_Face_Extraido
```

Listing 2: Importação dos dados - **Saída**.

A seguir, é realizada a listagem dos arquivos presentes no diretório de extração (caminho_extracao) após a extração do arquivo ZIP.

```
import os

# Listar arquivos no diretorio de extracao
os.listdir(caminho_destino)
```

Listing 3: Listagem dos dados extraídos.

```
['subject12.leftlight',
 'subject12.normal',
 'subject11.surprised',
 'subject14.glasses',
 'subject08.surprised',
 'subject13.centerlight',
 'subject15.rightlight',
 'subject13.rightlight',
 'subject07.sleepy',
 'subject12.surprised',
 'subject04.rightlight',
 'subject10.sad',
 'subject04.glasses',
```

```
'subject15.sleepy',
'subject11.sad',
'subject03.wink',
'subject07.centerlight',
'subject04.sleepy',
'subject07.normal',
'subject01.sad',
'subject14.surprised',
'subject02.leftlight',
'subject07.noglasses',
'subject03.surprised',
'subject11.sleepy',
'subject13.sleepy',
'subject04.normal',
'subject06.normal',
'subject03.happy',
'subject05.normal',
'subject07.surprised',
'subject12.glasses',
'subject02.wink',
'subject11.wink',
'subject12.centerlight',
'subject03.noglasses',
'subject02.sad',
'subject13.glasses',
'subject14.centerlight',
'subject11.normal',
'subject04.sad',
'subject10.normal',
'subject08.glasses',
'subject02.noglasses',
'subject13.sad',
'subject02.centerlight',
'subject15.happy',
'subject06.sleepy',
'subject14.wink',
'subject10.surprised',
'subject06.wink',
'subject12.sad',
'subject05.happy',
'subject10.glasses',
'subject01.normal',
'subject09.surprised',
'subject07.leftlight',
'subject05.sad',
'subject11.rightlight',
'subject07.happy',
'subject07.rightlight',
'subject09.centerlight',
'subject08.leftlight',
'subject05.wink',
'subject09.noglasses',
'subject09.normal',
'subject15.glasses',
'subject01.happy',
'subject05.glasses',
'subject09.rightlight',
'subject08.centerlight',
```

```
'subject01.wink',
'subject08.rightlight',
'subject12.sleepy',
'subject08.wink',
'subject02.glasses',
'subject07.glasses',
'subject02.sleepy',
'subject15.centerlight',
'subject01.noglasses',
'subject01.leftlight',
'subject05.rightlight',
'subject01.glasses',
'subject08.sleepy',
'subject10.leftlight',
'subject15.leftlight',
'subject04.happy',
'subject04.centerlight',
'subject05.leftlight',
'subject06.rightlight',
'subject06.glasses',
'subject05.noglasses',
'subject01.surprised',
'subject08.sad',
'subject10.happy',
'subject04.noglasses',
'subject06.happy',
'subject03.leftlight',
'subject11.glasses',
'subject02.surprised',
'subject11.happy',
'subject15.noglasses',
'subject08.happy',
'subject09.sad',
'subject13.surprised',
'subject14.sad',
'subject03.sleepy',
'subject06.noglasses',
'subject10.centerlight',
'subject06.surprised',
'subject06.centerlight',
'subject03.normal',
'subject11.leftlight',
'subject10.noglasses',
'subject09.sleepy',
'subject14.happy',
'subject01.rightlight',
'subject05.centerlight',
'subject03.sad',
'subject09.wink',
'subject15.normal',
'subject08.noglasses',
'subject12.happy',
'Readme.txt',
'subject04.wink',
'subject09.leftlight',
'subject10.wink',
'subject01.sleepy',
'subject14.sleepy',
```

```
 'data',
 'subject05.sleepy',
 'subject07.sad',
 'subject02.rightlight',
 'subject14.leftlight',
 'subject03.centerlight',
 'subject15.sad',
 'subject14.noglasses',
 'subject13.noglasses',
 'subject05.surprised',
 'subject03.rightlight',
 'subject09.glasses',
 'subject06.sad',
 'subject11.centerlight',
 'subject10.rightlight',
 'subject07.wink',
 'subject15.wink',
 'subject13.normal',
 'subject03.glasses',
 'subject12.noglasses',
 'subject14.normal',
 'subject13.leftlight',
 'subject01.centerlight',
 'subject02.normal',
 'subject12.wink',
 'subject04.leftlight',
 'subject13.wink',
 'subject09.happy',
 'subject11.noglasses',
 'subject08.normal',
 'subject12.rightlight',
 'subject14.rightlight',
 'subject15.surprised',
 'subject06.leftlight',
 'subject10.sleepy',
 'subject13.happy',
 'subject02.happy',
 'subject04.surprised']

```

Listing 4: Listagem dos dados extraídos - **Saída**.

A próxima etapa envolve a criação de um código para facilitar a visualização desses dados.

Para isso, a conversão de todos os arquivos da base de dados para o formato .jpg é necessária. Para essa finalidade, será utilizado o conteúdo da pasta "**data**" presente no banco de dados, que contém as informações a serem convertidas.

```
import os
from PIL import Image

def conversao():
    # Caminhos para os diretorios de entrada e saida
    caminho_dados = './Yale_Face_Extraido/data'
    caminho_dados_convertidos = './Pre_Processo'

```

```
    # Verifica se o diretorio de saida existe; se nao, cria-o
    if not os.path.isdir(caminho_dados_convertidos):
        os.mkdir(caminho_dados_convertidos)

        # Itera pelos arquivos no diretorio de entrada
        for nome_do_arquivo in os.listdir(caminho_dados):
            # Verifica se o arquivo nao e do tipo gif, txt ou DS_Store
            if not 'gif' in nome_do_arquivo and not 'txt' in
nome_do_arquivo and not 'DS_Store' in nome_do_arquivo:
                # Abre a imagem
                imagem = Image.open(caminho_dados + '/' +
nome_do_arquivo)
                # Obtem o numero do sujeito da imagem
                nome_da_classe = nome_do_arquivo.split('.')[1]
                # Define o caminho para a imagem convertida
                caminho_da_imagem = caminho_dados_convertidos + '/' +
nome_da_classe + '/' + nome_do_arquivo + '.jpg'

                # Verifica se o diretorio para o sujeito existe; se nao,
 cria-o
                if not os.path.isdir(caminho_dados_convertidos + '/' +
nome_da_classe):
                    os.mkdir(caminho_dados_convertidos + '/' +
nome_da_classe)

                # Verifica se o arquivo convertido nao existe; se nao, o
 salva
                if not os.path.isfile(caminho_da_imagem):
                    imagem.save(caminho_da_imagem)
    else:
        print("Procedimento Concluido!")
```

Listing 5: Conversão dos dados extraídos.

Em seguida, executa-se a função "**conversao()**", responsável por converter o tipo dos arquivos pertencentes à base de dados.

```
conversao()
```

Listing 6: Aplicação da função "conversao()".

Por fim, o próximo conjunto de código gera a visualização de uma amostra específica da base de dados em estudo. Essa seleção inclui as imagens das 15 pessoas piscando, correspondentes aos arquivos associados à classe "**wink**".

É importante destacar que, na abordagem adotada, os exemplos de expressões faciais foram categorizados como classes do sistema, resultando em um total de 11 padrões de classe.

```
import os
import matplotlib.pyplot as plt
import cv2 as cv

# Caminho para a pasta que contem as imagens
caminho_pasta = './Pre_Processo/wink'

```

```
# Obter a lista de arquivos na pasta
arquivos = os.listdir(caminho_pasta)

# Configuracao da plotagem das amostras desejadas
num_linhas = 5
num_colunas = 3
fig, axs = plt.subplots(num_linhas, num_colunas, figsize=(10, 15))

# Iterar sobre os arquivos e exibir as imagens
for i, arquivo in enumerate(arquivos):
    caminho_imagem = os.path.join(caminho_pasta, arquivo)
    img = cv.imread(caminho_imagem)
    axs[i // num_colunas, i % num_colunas].imshow(cv.cvtColor(img, cv.COLOR_BGR2RGB))
    axs[i // num_colunas, i % num_colunas].axis('off')  # Desativar os eixos

plt.show()
```

Listing 7: Visualização dos dados convertidos.

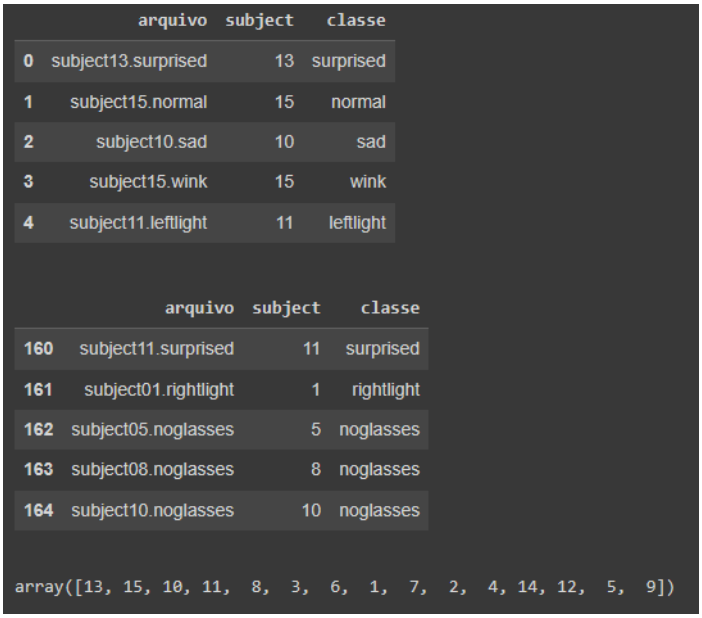
Figura 4: Visualização dos dados convertidos - **Saída**.

Separação dos Dados para Treinamento e Teste:

Criação de um dataframe contendo o nome do arquivo, “subject” (número da pessoa) e classe.

```
import pandas as pd
import os

# Caminho dos dados pela pasta 'data'
caminho_dados = './Yale_Face_Extraido/data'

# Listar todos os arquivos no diretorio
arquivos = os.listdir(caminho_dados)

# Filtrar apenas os arquivos que comecam com 'subject'
arquivos_filtrados = [arq for arq in arquivos if arq.startswith('subject')]

# Criacao do DataFrame com os nomes dos arquivos filtrados
dados = pd.DataFrame(arquivos_filtrados, columns=['arquivo'])

# Separar os componentes do nome do arquivo
dados[['subject', 'classe']] = dados['arquivo'].str.split(pat=".", n=1, expand=True)
dados['subject'] = dados['subject'].str.replace('subject', '', regex=False)

# Remover linhas com valores ausentes (NaN)
dados.dropna(subset=['subject'], inplace=True)

# Converter a coluna 'subject' para valores inteiros
dados['subject'] = dados['subject'].astype(int)

# Mostrar o DataFrame resultante
display(dados.head())
print('\n')

display(dados.tail())
print('\n')

display(dados['subject'].unique())
```

Listing 8: Criação do DataFrame.



	arquivo	subject	classe
0	subject13.surprised	13	surprised
1	subject15.normal	15	normal
2	subject10.sad	10	sad
3	subject15.wink	15	wink
4	subject11.leftlight	11	leftlight

	arquivo	subject	classe
160	subject11.surprised	11	surprised
161	subject01.rightlight	1	rightlight
162	subject05.noglasses	5	noglasses
163	subject08.noglasses	8	noglasses
164	subject10.noglasses	10	noglasses

```
array([13, 15, 10, 11,  8,  3,  6,  1,  7,  2,  4, 14, 12,  5,  9])
```

Figura 5: Exibição do Head e Tail do DataFrame gerado - **Saída**.

Nessa perspectiva, executa-se a divisão entre os dados, os quais serão fornecidos à máquina futuramente.

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

total_de_classes = 11
total_de_imagens = dados["subject"].count()
expressoes_faciais = ['centerlight', 'glasses', 'happy', 'leflight', '
    noglasses', 'normal', 'rightlight', 'sad', 'sleepy', 'surprised', '
    wink']

y = dados['classe']

# Aplicar codificacao one-hot na coluna 'classe'
encoded_y = pd.get_dummies(dados['classe'])

# Separar 'X' sem a coluna 'classe' e 'arquivo', uma vez que sao
    compostos por strings
X = dados.drop('classe', axis=1)
X1 = X.drop('arquivo', axis =1)

# Normalizacao dos dados
scaler = MinMaxScaler()
Xn = scaler.fit_transform(X1)

# Unir 'Xn' (dados normalizados) com as expressoes faciais codificadas
X_encoded = np.concatenate((Xn, encoded_y), axis=1)

# Separar os dados para treino e teste
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y,
    test_size=0.4, stratify=y)
```

Listing 9: Separação dos dados.

Criação da MLP:

O tópico 3 traz o desenvolvimento da MLP, usando a biblioteca **scikit-learn**, tendo em vista a praticidade que esta ferramenta detém.

Posteriormente, ocorre a validação e análise do modelo, sendo feita a plotagem das matrizes confusões com o uso da biblioteca **seaborn** e **matplotlib**, bem como o valor da média entre as acurácias do sistema por meio da própria scikit-learn.

```
import time
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.exceptions import ConvergenceWarning
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
    classification_report
```

```
11
12 # Numero de vezes que voce deseja repetir o treinamento para calcular a
       media
13 num_execucoes = 5
14 acuracias = []
15 tempo_treinos = []
16
17 # Desativar os warnings de convergencia temporariamente
18 warnings.filterwarnings("ignore", category=ConvergenceWarning)
19
20 for n in range(num_execucoes):
21
22     # Passo 1: Criacao do Modelo MLP
23     mlp = MLPClassifier(hidden_layer_sizes=(32, 16), activation='relu',
       max_iter=100)
24
25     # Passo 2: Treinamento do Modelo
26     inicio_tempo = time.time()
27     mlp.fit(X_train, y_train)
28     final_tempo = time.time()
29     tempo_treino = final_tempo - inicio_tempo
30     tempo_treinos.append(tempo_treino)
31
32     # Passo 3: Avaliacao da Acuracia
33     y_pred = mlp.predict(X_test)
34     acc = accuracy_score(y_test, y_pred)
35     acuracias.append(acc)
36
37     # Passo 4: Criacao da matriz confusao
38     cm = confusion_matrix(y_test, y_pred)
39
40     plt.figure(figsize=(8, 6))
41     sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', annot_kws={"size"
       : 14}, xticklabels=expressoes_faciais, yticklabels=expressoes_faciais
       )
42
43     plt.xlabel('Previsoes')
44     plt.ylabel('Valores Verdadeiros')
45     plt.title(f'Matriz de Confusao {n+1}')
46     plt.show()
47     print('\n')
48
49 # Calculando a media das acuracias e do tempo de treinamento
50 media_acuracias = sum(acuracias) / num_execucoes
51 media_tempo_treino = sum(tempo_treinos) / num_execucoes
52
53 # Exibindo resultado da media das acuracias e tempos de treinamentos
54 print("Media da Acuracia do Modelo MLP:", media_acuracias)
55 print("Media do Tempo de Treinamento:", media_tempo_treino, "segundos")
56 print('\n')
57
58 # Exibindo relatorio de classificacao da quinta iteracao e/ou execucao
59 relatorio_de_classificacao = classification_report(y_test, y_pred,
       target_names=expressoes_faciais)
60 print(relatorio_de_classificacao)
```

Listing 10: Desenvolvimento da Rede Neural de Múltiplas Camadas.

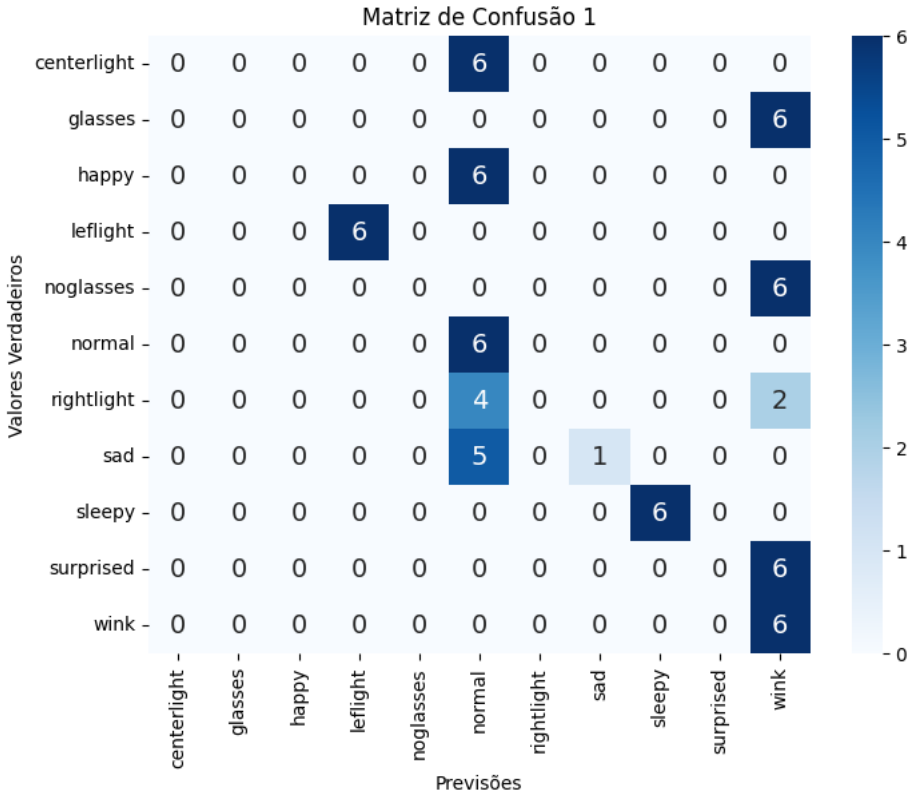


Figura 6: Matriz confusão da primeira iteração - **Saída**.

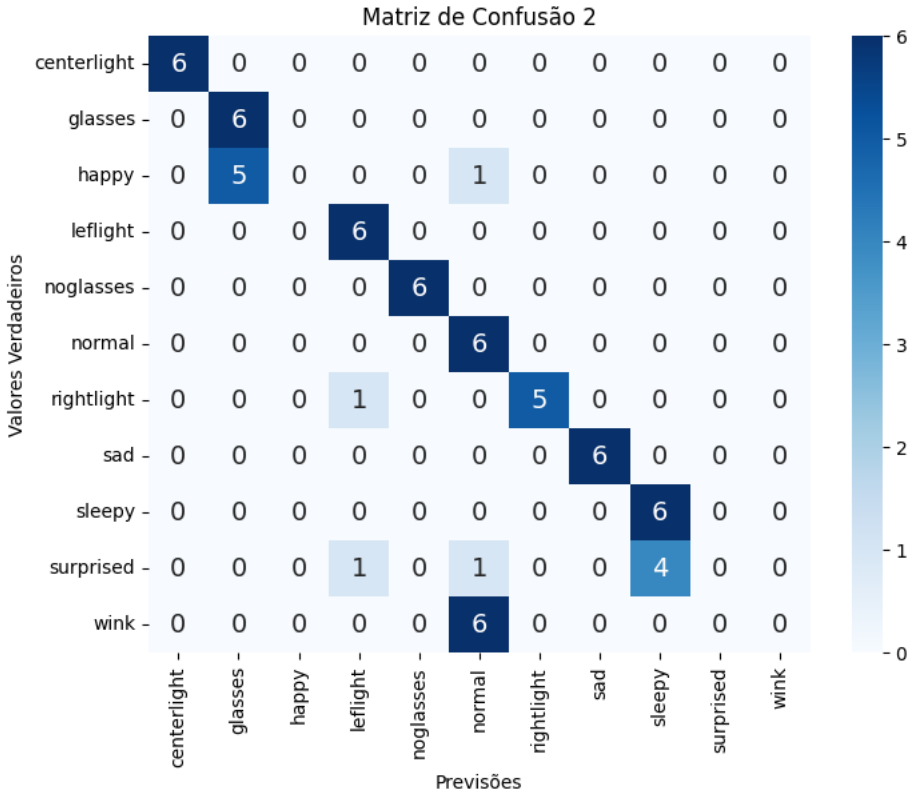


Figura 7: Matriz confusão da segunda iteração - **Saída**.

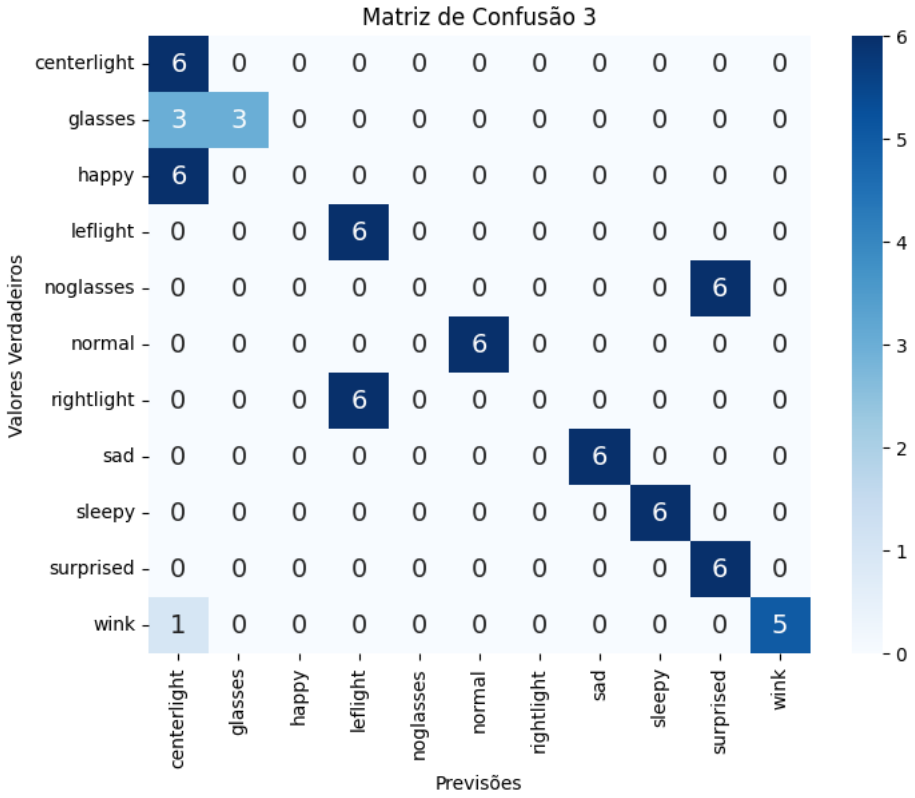


Figura 8: Matriz confusão da terceira iteração - **Saída**.

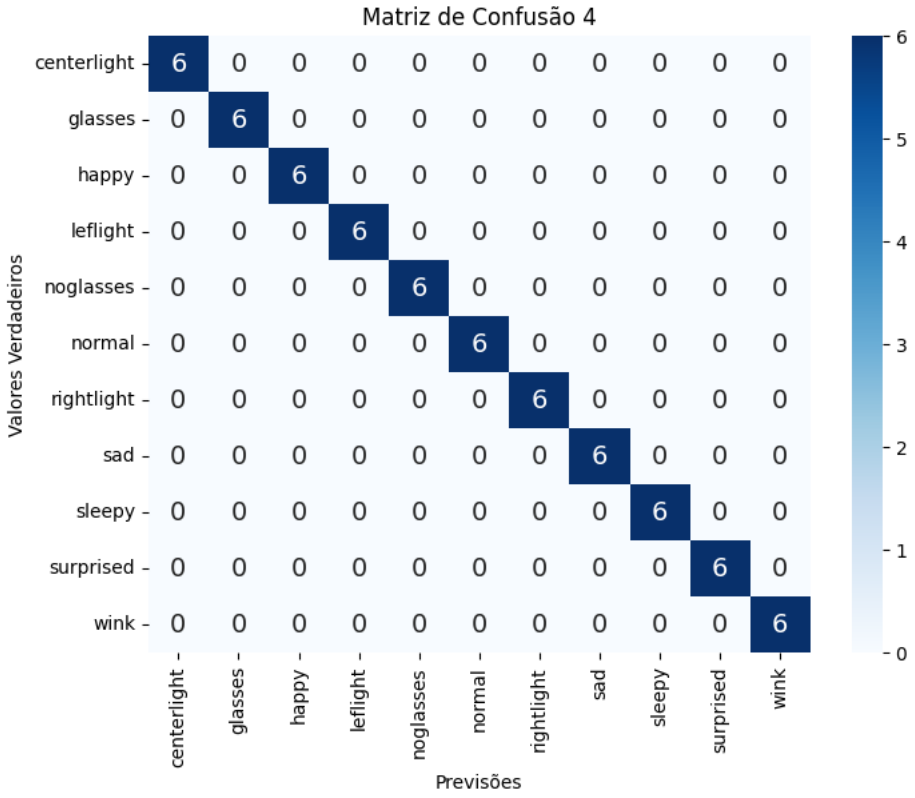


Figura 9: Matriz confusão da quarta iteração - **Saída**.

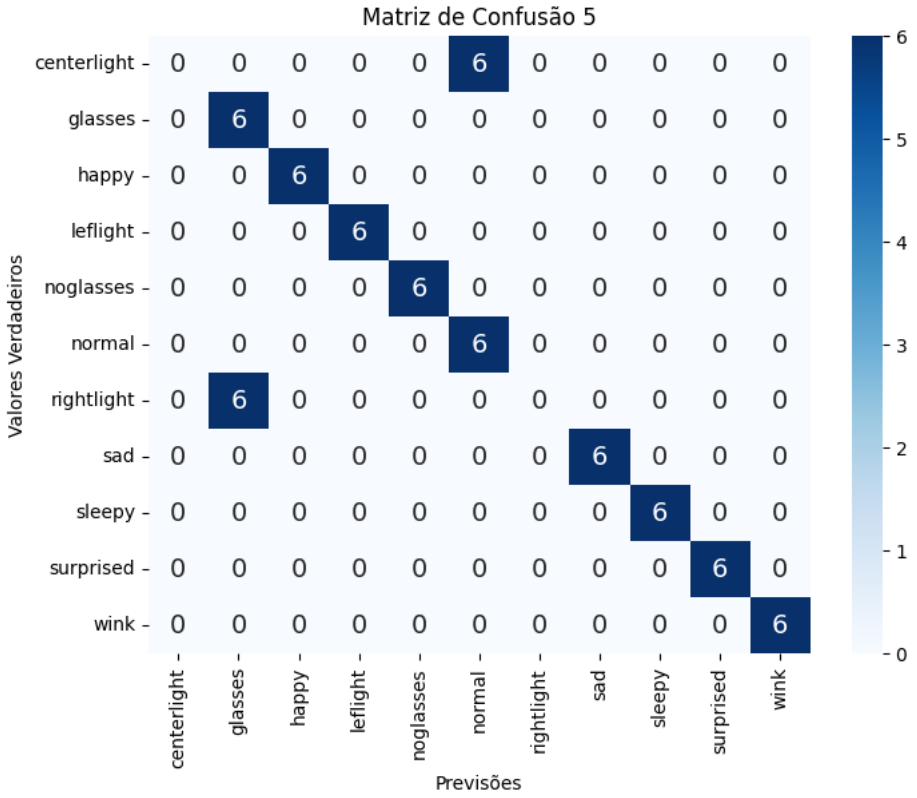


Figura 10: Matriz confusão da quinta iteração - **Saída**.

Nesse sentido, vale ressaltar que a quarta matriz de confusão, exibida pela figura 9, apresentou apenas acertos, ou seja, a máquina obteve 100% de aproveitamento.

```
Media da Acuracia do Modelo MLP: 0.7151515151515151
Media do Tempo de Treinamento: 0.13404459953308107 segundos


              precision    recall  f1-score   support

 centerlight       0.00      0.00      0.00         6
     glasses       0.50      1.00      0.67         6
       happy       1.00      1.00      1.00         6
    leflight       1.00      1.00      1.00         6
   noglasses       1.00      1.00      1.00         6
      normal       0.50      1.00      0.67         6
  rightlight       0.00      0.00      0.00         6
         sad       1.00      1.00      1.00         6
      sleepy       1.00      1.00      1.00         6
   surprised       1.00      1.00      1.00         6
        wink       1.00      1.00      1.00         6

    accuracy                           0.82        66
   macro avg       0.73      0.82      0.76        66
weighted avg       0.73      0.82      0.76        66

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.
   py:1344: UndefinedMetricWarning: Precision and F-score are ill-
   defined and being set to 0.0 in labels with no predicted samples. Use
    'zero_division' parameter to control this behavior.
```

```
24   _warn_prf(average, modifier, msg_start, len(result))
25 /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.
     py:1344: UndefinedMetricWarning: Precision and F-score are ill-
     defined and being set to 0.0 in labels with no predicted samples. Use
      'zero_division' parameter to control this behavior.
26   _warn_prf(average, modifier, msg_start, len(result))
27 /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.
     py:1344: UndefinedMetricWarning: Precision and F-score are ill-
     defined and being set to 0.0 in labels with no predicted samples. Use
      'zero_division' parameter to control this behavior.
28   _warn_prf(average, modifier, msg_start, len(result))
```

Listing 11: Média das acurácias e tempos de treinamento - **Saída**.

3 Conclusão

A abordagem completa do ciclo de processamento e análise da Yale Face Database, desde a exploração dos conceitos fundamentais das redes neurais até a sua implementação prática em Python, revelou contribuições substanciais para a compreensão e manipulação dos dados.

Finalmente, ao entender profundamente a estrutura e as nuances dessa base de dados, nosso projeto não apenas ofereceu uma visão holística das técnicas de processamento de imagens e redes neurais, mas também estabeleceu um alicerce sólido para investigações futuras e aplicações práticas na identificação e reconhecimento de padrões faciais.

4 Referências

[1] **Yale Face Database**. Disponível em: https://www.kaggle.com/datasets/olgabelitskaya/yale-face-database. Acesso em: 28 de novembro de 2023.

[2] MOREIRA, S. **Rede Neural Perceptron Multicamadas**. Disponível em: https://medium.com/ensina-ai/rede-neural-perceptron-multicamadas-f9de8471f1a9#:~:text=Perceptron%20Multicamadas%20%28PMC%20ou%20MLP%20%E2%80%94%20Multi%20Layer. Acesso em: 30 de novembro de 2023.

[3] OPENAI. **ChatGPT**. 2023. Disponível em: https://openai.com/. Acesso em: 30 de novembro de 2023.