

Aprendizagem de Máquina (2020/Período Especial) - Redes Neurais

Diogo C. T. Batista¹

¹Universidade Federal do Paraná (UFPR)
Curitiba – Paraná – Brasil

diogo@diogocezar.com

1. Introdução

Neste laboratório, foi considerada uma base de dados com os meses do ano. As imagens são representações manuscritas das palavras que compreendem os meses: Janeiro, Fevereiro, Março, Abril, Maio, Junho, Julho, Agosto, Setembro, Outubro, Novembro e Dezembro. Portanto, são 12 classes. A Figura 1 demonstra alguns exemplos dessas imagens.



Figura 1. Exemplos dos meses manuscritos

As implementações para este laboratório foram realizadas na plataforma Google Colab. Foram realizadas as seguintes etapas: Preparações, Definição dos Modelos, Execução do Modelo LeNet5 sem Data Augmentation, Execução de um Modelo Personalizado sem Data Augmentation, Implementação de Data Augmentation, Execução do Modelo LeNet5 com Data Augmentation, Execução de um Modelo Personalizado com Data Augmentation, Extração de Características e Implementação do SVM com as Características Extraídas.

Na sequência cada uma das etapas será detalhada, seguido da análise dos resultados obtidos em cada uma das etapas, bem como as suas comparações.

Para a execução dos experimentos, utilizou-se os Frameworks Keras e Sklearn.

2. Preparações

Nesta etapa do laboratório foram realizadas as preparações para os experimentos: Habilitação da GPU, Importação dos dados hospedados no GitHub, Definição dos Arquivos de Entrada e Definição das Funções Auxiliares.

Com a execução dos experimentos foi realizado no Google Colab, os experimentos foram realizados com a habilitação do processamento via GPU.

Na definição das funções auxiliares, a ideia foi definir possíveis componentes reutilizáveis nos experimentos, encapsulando implementações em funções reutilizáveis.

3. Definição dos Modelos

Definiu-se também os modelos como funções. O primeiro modelo utilizado foi o LeNet 5, suas camadas estão representadas na Figura 2

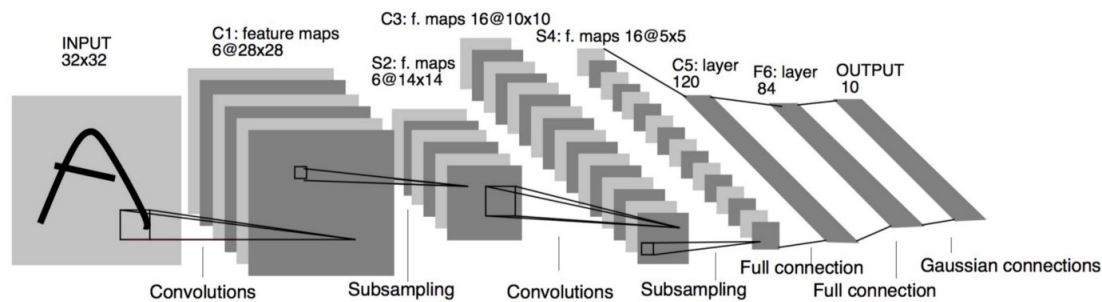


Figura 2. Exemplos das camadas utilizadas pelo modelo LeNet5

O segundo modelo, é o mesmo utilizado nas demonstrações em aula e implementa as seguintes camadas detalhadas no Código 1.

```

1 model = Sequential()
2 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(img_rows,
   img_cols, 3)))
3 model.add(Conv2D(64, (3, 3), activation='relu'))
4 model.add(MaxPooling2D(pool_size=(2, 2)))
5 model.add(Dropout(0.25))
6 model.add(Flatten())
7 model.add(Dense(128, activation='relu'))
8 model.add(Dropout(0.5))
9 model.add(Dense(num_classes, activation='softmax'))

```

Código 1. Modelo Personalizado

4. Execução do Modelo LeNet5 sem Data Augmentation

Após todas as preparações, o primeiro experimento realizado foi a execução de uma rede utilizando o modelo LeNet5, com os dados originais, sem qualquer tipo de ajuste.

Quanto ao tamanho das imagens de entrada, utilizou-se o tamanho de 32 por 32 pixels, assim como o modelo prevê.

Os números de épocas, utilizou-se experimentos com as seguintes variações: 32, 64 e 128.

Também se mediu os tempos de execuções em cada um dos experimentos.

Os passos para a execução foram:

1. Carregamento inicial dos dados (x_train, y_train, x_test, y_test). Neste passo uma função genérica faz os ajustes necessários nas imagens e retorna os vetores;
2. Normalização das imagens;
3. Geração dos Labels para a matrix de confusão;
4. Conversão dos vetores para matrizes binárias;
5. Definição e Sumarização do modelo;
6. Compilação do modelo;
7. Treinamento da Rede;
8. Obtenção dos resultados.

Após a execução, foi possível observar os resultados:

5. Execução de um Modelo Personalizado sem Data Augmentation

Na sequência repetiu-se os mesmos passos, mas neste caso, com outro modelo de treinamento.

Os resultados obtidos foram:

6. Implementação de Data Augmentation

Neste passo do experimento, foram implementadas técnicas para a realização de Data Augmentation. Para isso, foram implementadas funções específicas para a realização de pequenas variações nas imagens da base de treinamento original. O Código 2 mostra quais foram as funções utilizadas para a realização destas transformações.

```
1 flip_rotation_brightness_zoom(path, zoom=[0.5, 1.0], brightness=[0.2, 1.0], rotation
    =90, flip_horizontal=False,
2 flip_vertical=False, subdir="zoom")
3
4 random_zoom(path, zoom=[0.5, 1.0], subdir="zoom")
5
6 random_brightness(path, brightness=[0.2, 1.0], subdir="brightness")
7
8 random_rotation(path, rotation=90, subdir="rotation")
9
10 horizontal_vertical_flip(path, flip_horizontal=False, flip_vertical=False, subdir="
    flip")
11
12 orizontal_vertical_shift(path, size=0.5, bool_width=True, subdir="shift")
```

Código 2. Funções de Transformação das Imagens

A função **flip_rotation_brightness_zoom** realiza transformações de rotação, brilho e zoom na imagem. A função **random_zoom** realiza a transformação de zoom da imagem. A função **random_brightness** realiza a transformação de brilho da imagem. A função **random_rotation** realiza a transformação de rotação da imagem. A função **horizontal_vertical_flip** realiza a transformação de flip vertical. A função **horizontal_vertical_shift** realiza a transformação de shift. Todas as funções possuem parâmetros.

Para imagem da base de treinamento foram realizadas as operações descritas no Código 3

```
1 horizontal_vertical_flip(image_path, flip_horizontal=True, flip_vertical=False)
2 horizontal_vertical_flip(image_path, flip_horizontal=False, flip_vertical=True)
3 horizontal_vertical_flip(image_path, flip_horizontal=True, flip_vertical=True)
4 horizontal_vertical_flip(image_path, flip_horizontal=False, flip_vertical=False)
5 horizontal_vertical_shift(image_path, bool_width=True)
6 horizontal_vertical_shift(image_path, bool_width=False)
7 random_rotation(image_path, rotation=10)
8 random_rotation(image_path, rotation=20)
9 random_rotation(image_path, rotation=30)
10 random_rotation(image_path, rotation=45)
11 random_brightness(image_path)
12 random_brightness(image_path, brightness=[0.1, 0.2])
13 random_brightness(image_path, brightness=[0.3, 0.4])
14 random_brightness(image_path, brightness=[0.4, 0.5])
15 random_zoom(image_path)
16 random_zoom(image_path, zoom=[0.1, 0.5])
17 random_zoom(image_path, zoom=[0.1, 0.2])
18 random_zoom(image_path, zoom=[0.1, 0.3])
19 flip_rotation_brightness_zoom(image_path, rotation=30)
20 flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.5], brightness=[0.1, 0.5])
```

```

21 flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.5], brightness=[0.1, 0.5])
22 flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.5], brightness=[0.1, 0.5],
    rotation=30)
23 flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.5], brightness=[0.1, 0.5],
    rotation=30)
24 flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.5], brightness=[0.1, 0.5],
    rotation=30)
25 flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.8], brightness=[0.1, 0.8],
    rotation=45)
26 flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.8], brightness=[0.1, 0.8],
    rotation=45)
27 flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.2], brightness=[0.1, 0.2],
    rotation=30)
28 flip_rotation_brightness_zoom(image_path, zoom=[0.1, 0.2], brightness=[0.1, 0.2],
    rotation=45)
29 flip_rotation_brightness_zoom(image_path, zoom=[0.9, 1], brightness=[0.9, 1],
    rotation=30)
30 flip_rotation_brightness_zoom(image_path, zoom=[0.9, 1], brightness=[0.9, 1],
    rotation=45)

```

Código 3. Funções Executadas

A base inicial de treinamento, é composta por 1578 imagens. Após execução do script, obteve-se o total de X imagens.

7. Execução do Modelo LeNet5 com Data Augmentation

Na sequência, foram utilizados os novos dados para nova execução do modelo LeNet5.

Quanto ao tamanho das imagens de entrada, utilizou-se o mesmo tamanho de 32 por 32 pixels, assim como o modelo prevê.

Os números de épocas, utilizou-se experimentos com as seguintes variações: 32, 64 e 128.

Também se mediu os tempos de execuções em cada um dos experimentos.

Dessa vez, os resultados obtidos foram:

8. Execução de um Modelo Personalizado com Data Augmentation

Na sequência repetiu-se os mesmos passos, mas neste caso, com outro modelo de treinamento.

Os resultados obtidos foram:

9. Extração de Características

Na sequência dos experimentos, utilizou-se um modelo pré-treinado da imagenet para a extração das características das imagens, como demonstrado no Código 4.

```

1 model = InceptionV3(weights='imagenet', include_top=False)

```

Código 4. ImageNet - InceptionV3

Para isso, criou-se um arquivo no formato SVM como no exemplo descrito no Código 5.

```

1 0 1:0.000000 2:0.000000 3:0.001412 4:0.000000 5:0.014124 6:0.000000 ...

```

Código 5. Exemplo do Formato de Entrada

10. Implementação do SVM com as Características Extraídas

Utilizando os arquivos gerados de treinamento e teste, aplicou-se o classificador SVM, obtendo os resultados:

11. Discussão dos Resultados