UNIVERSIDADE FEDERAL DO PARANÁ

SETOR DE TECNOLOGIA

CURSO DE INFORMÁTICA BIOMÉDICA

ANNA CAROLINE BOZZI

CNN

Curitiba
2020

1. Introdução

Esse trabalho apresenta os resultados da classificação de uma base de imagens, com manuscrito referente aos doze meses do ano, utilizando duas redes neurais convolucionais:

- *CNN*
- *LeNet5*

Foi feita a implementação em python de ambas, e também de técnicas para aumentar a base de treinamento, *Data Augmentation.*

A base de dados consistia em 1979 imagens, das quais 1578 foram pré determinadas para treinamento e 401 para teste.

Foi ainda utilizando duas redes pré-treinadas da *ImageNet, Transfer Learning* e o *Fine-Tuning,* para gerar vetores de características para classificar em um SVM.

2. Implementações:

2.1 CNN:

Carregamento de dados

Para o carregamentos dos dados foi utilizado o *Keras*.

Pré processamento

Para esse passo foi utilizado as funções de *resize* e *reshape*.

Particionamento

O particionamento de dados foi determinado em 80% para treinamento e 20% para teste/validação, e ainda foi realizado teste com técnicas de Data Augmentation para aumento da base de treinamento.

Rede

São três camadas convolucionais. Para as primeiras camadas da rede utiliza-se *Conv2D().* Em seguida é adicionado a camada de pooling máximo *MaxPooling2D().* É utilizado também *Dropout()* para evitar *overfitting*.

Treinamento

Assim que compilado o modelo, é realizado o treinamento utilizando a função *fit()* do *Keras*.

2.2 LeNte 5

Carregamento de dados

Para o carregamentos dos dados foi utilizado o *Keras*.

Pré processamento

Para esse passo foi utilizado as funções de *resize* e *reshape*.

Particionamento

O particionamento de dados foi determinado em 80% para treinamento e 20% para teste/validação, e ainda foi realizado teste com técnicas de Data Augmentation para aumento da base de treinamento.

Rede

A diferença do CNN é nas camadas, para esse modelo foram 5 camadas de convoluções.

Treinamento

Assim que compilado o modelo, é realizado o treinamento utilizando a função *fit()* do *Keras*.

2.3 Data Augmentation

Foram utilizadas três técnicas combinadas para o aumento da base de treinamento:

- Rotação, em 8 graus apenas
- Zoom de 20%
- e Variações de brilho entre 0.5 e 1.5

A rotação faz com que a imagem seja levemente inclinada para ambos os lados. O zoom foi aplicado centralizado de forma que não houvesse “corte” na palavra. E o brilho foi aplicado de forma que apresentasse pequena variação na intensidade dos pixels.

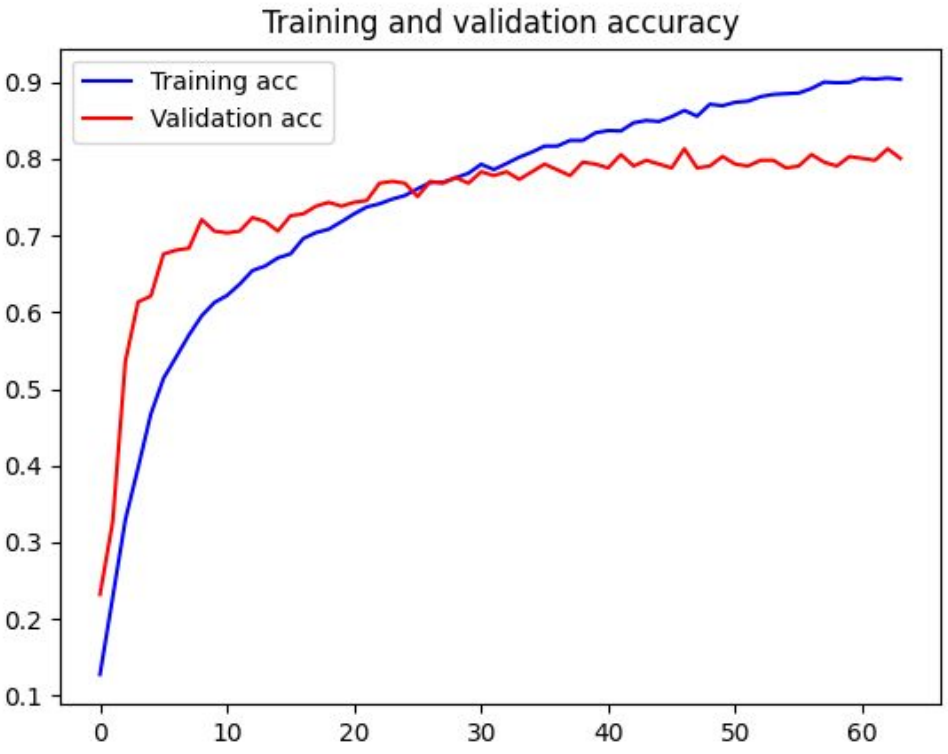
Com essas técnicas foram geradas 3 novas imagens para cada uma disponível para teste. Ou seja, no total a base de treinamento com Data Augmentation foi de 6312 imagens

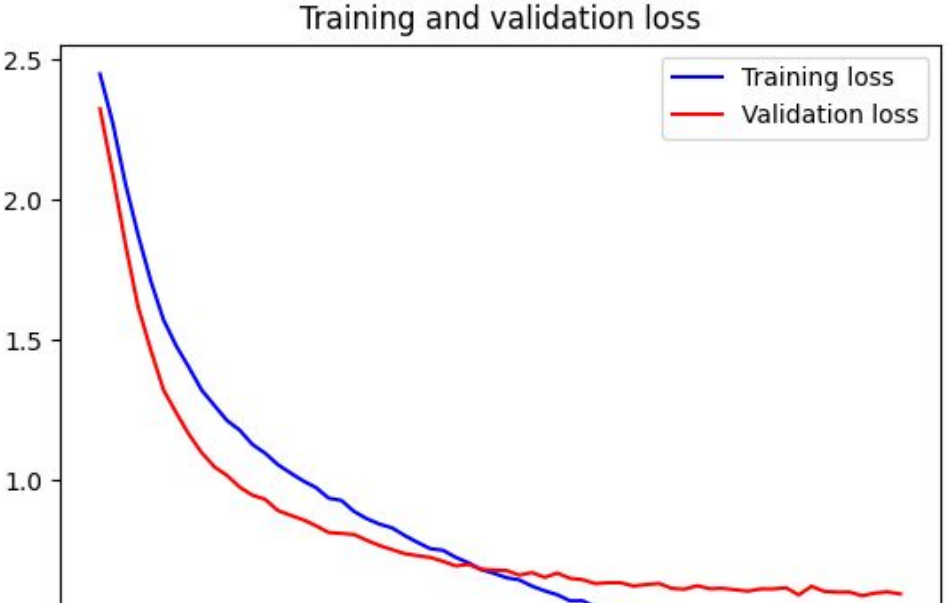
3.Resultados e análise

3.1 CNN

3.1.1 Com Data Augmentation, epochs = 65, learning_rate = 0.01, batch = 64.

É possível ver nos gráficos a seguir que a *validation loss* e a *validation accuracy* estão sincronizadas com a *training loss* e *training accuracy*. Mesmo que as linhas de *accuracy* não sejam lineares, mostra que **o modelo por volta da época 30 não está em *overfitting,*** a *validation loss* está diminuindo e não aumentando, e a lacuna entre a *accuracy* do *training* e da *validation* é sutil, porém a partir da época 30 o modelo dirige-se a um *overfitting*. O ideal, observando esse caso, seria ter parado por volta da época 30.


Training and validation accuracy
Training acc
Validation acc
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50
60

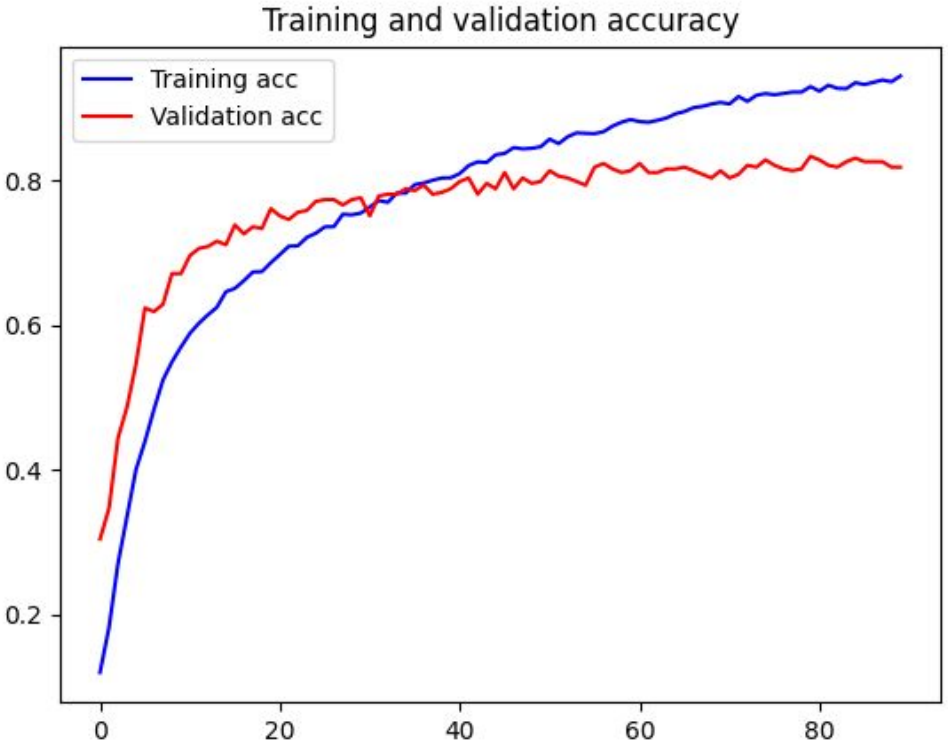

Training and validation loss
Training loss
Validation loss
2.5
2.0
1.5
1.0

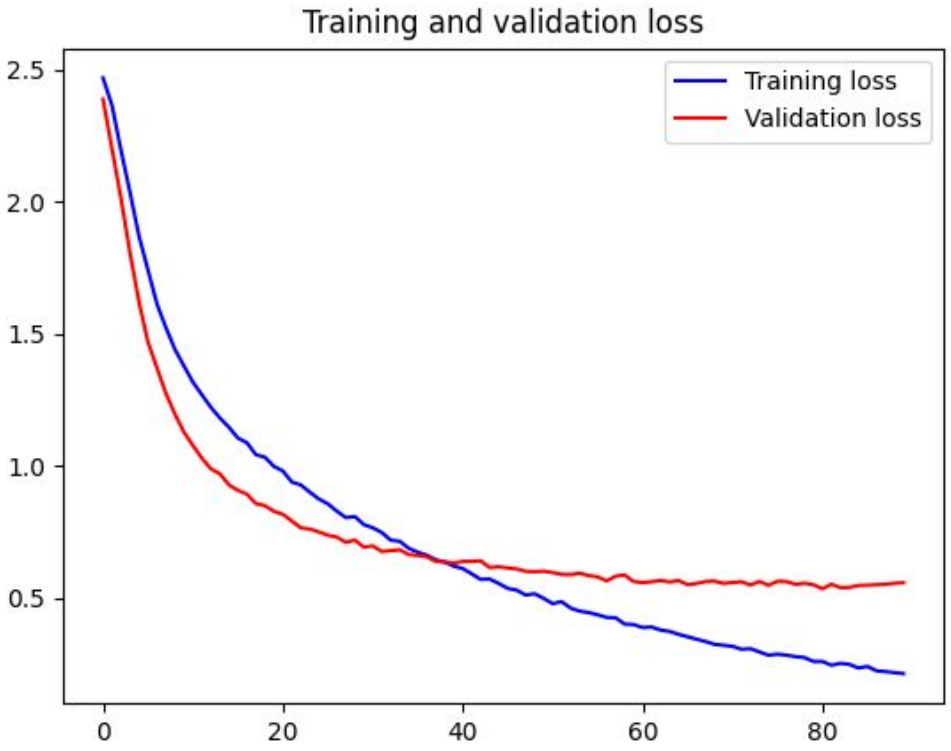
```
[[28  5  0  0  0  2  3  0  0  0  0  1]
 [ 4 25  0  0  0  0  0  0  1  2  0  0]
 [ 0  0 35  1  0  0  0  0  0  0  0  0]
 [ 0  0  1 32  4  0  0  1  1  0  0  0]
 [ 0  0  1  3 34  0  0  0  0  0  0  0]
 [ 7  0  0  0  0 19  2  1  0  0  0  0]
 [ 5  0  0  0  2  0 25  0  0  0  0  0]
 [ 0  0  0  0  0  1  0 24  0  0  1  2]
 [ 0  0  0  1  0  0  0  0 20  5  2  3]
 [ 0  0  1  1  0  0  0  0  0 28  0  0]
 [ 0  0  0  0  0  0  0  0  1  2 31  0]
 [ 0  0  0  0  0  2  0  2  4  3  2 20]]
```

A matriz de confusão para esse classificador demonstra bons resultado em relação às classificações, é possível verificar confusão relativamente baixa. A frequência maior de confusão é entre os meses Janeiro Junho e Julho.

3.1.2 Com Data Augmentation, epochs = 90, learning_rate= 0.02, batch = 64.

Conforme foi demonstrado, a *CNN* para 60 épocas apresentou uma tendência a *overfitting*, realizado agora o teste para 90 épocas é possível afirmar que a partir de 30 épocas há *overfitting*.



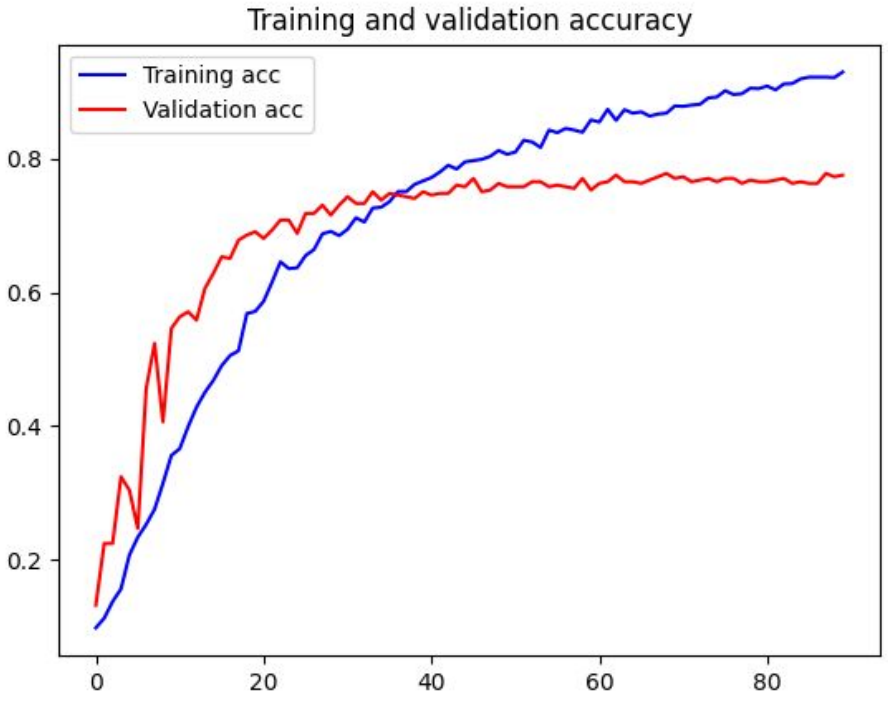


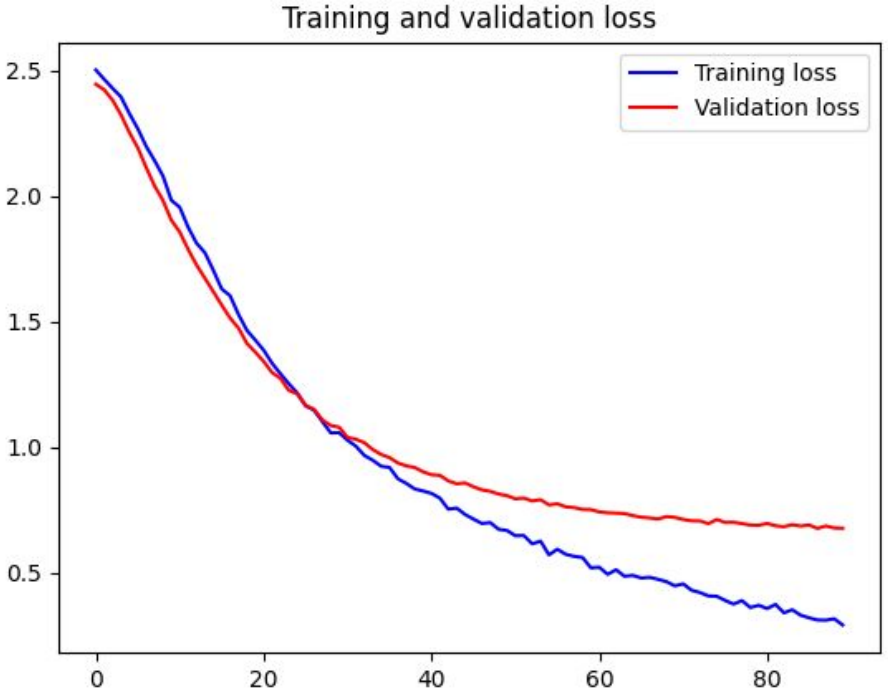

```
[[31  2  0  0  0  2  3  0  0  0  0  1]
 [ 5 22  0  1  0  0  0  0  1  0  3  0]
 [ 0  0 35  1  0  0  0  0  0  0  0  0]
 [ 0  0  0 33  2  0  0  2  2  0  0  0]
 [ 0  1  0  4 33  0  0  0  0  0  0  0]
 [ 7  0  0  0  0 20  1  1  0  0  0  0]
 [ 4  0  0  0  2  3 23  0  0  0  0  0]
 [ 0  0  0  0  0  1  0 25  0  0  1  1]
 [ 0  0  0  1  0  0  0  0 25  2  2  1]
 [ 0  0  0  1  0  0  0  0  0 29  0  0]
 [ 0  0  0  0  0  0  0  0  1  1 32  0]
 [ 0  0  0  0  0  2  0  1  5  2  3 20]]
```

A respectiva matriz de confusão dessa execução demonstra pouca variação em relação a com 60 épocas. Apresenta as mesmas confusões mais aparentes, entre os meses de janeiro, junho e julho.

3.1.2 Sem Data Augmentation, epochs = 90, learning_rate = 0.02, batch = 64.

Como foi visto anteriormente, com 60 épocas a classificação com Data Augmentation apresentou algumas dúvidas, será portanto apresentado de forma direta a classificação sem Data Augmentation para 90 épocas. Conforme é possível verificar abaixo nos gráficos, o mesmo desempenho geral pode ser observado, também a partir de 30 épocas há uma tendência a *overfitting*, salientando que seria o ideal parar.






```
[[30  5  0  0  0  2  1  0  0  0  0  1]
 [ 7 20  0  1  0  0  0  0  1  2  1  0]
 [ 0  0 34  1  0  0  0  1  0  0  0  0]
 [ 0  0  0 35  0  0  0  2  2  0  0  0]
 [ 1  0  0  2 34  0  1  0  0  0  0  0]
 [ 7  1  0  0  0 18  2  1  0  0  0  0]
 [ 7  0  0  0  0  3 21  1  0  0  0  0]
 [ 0  0  1  1  0  0  0 23  0  0  1  2]
 [ 0  0  0  1  1  0  0  0 25  1  2  1]
 [ 0  0  2  0  1  0  0  0  1 26  0  0]
 [ 0  0  0  1  1  0  0  0  3  2 27  0]
 [ 1  2  0  1  0  1  0  2  6  2  0 18]]
```

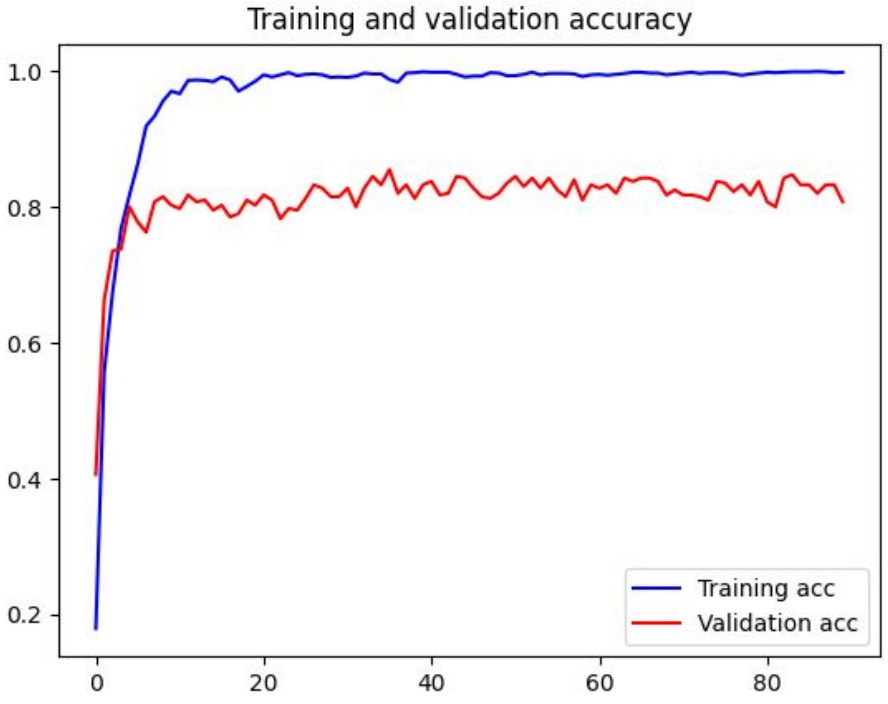
A matriz de confusão resultante apresenta no geral as mesmas confusões das já demonstradas pela classificação com Data Augmentation.

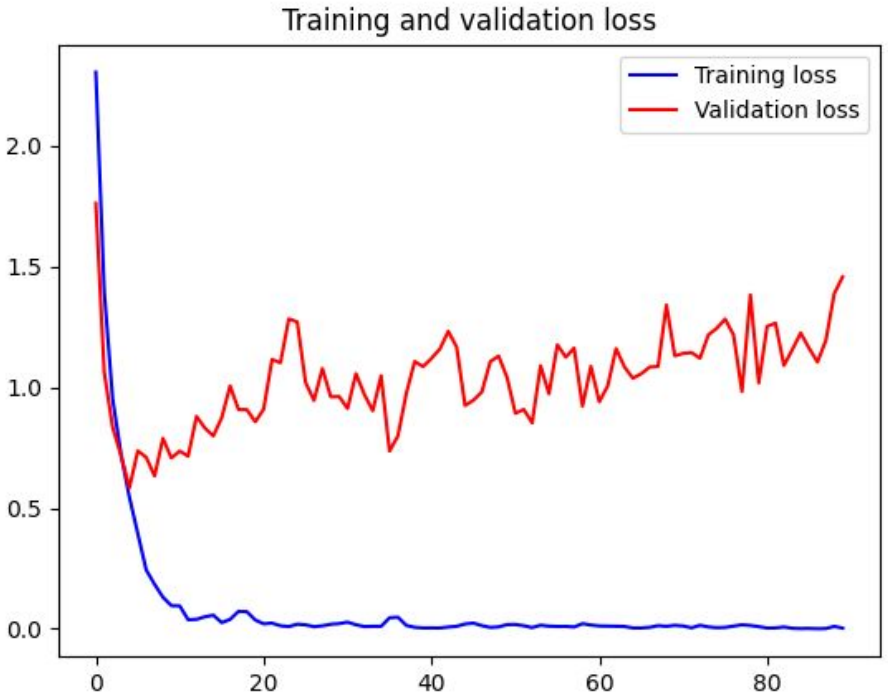
3.2 LeNet5

Como os testes de classificação feito previamente com o CNN indicaram que um maior número de épocas seria o ideal para apresentar uma análise mais precisa, foi iniciado diretamente esse classificador para 90 épocas.

3.2.1 Com Data Augmentation, epochs = 90, learning_rate = 0.01 , batch = 64.

É possível ver nos gráficos a seguir que a *validation loss* e a *validation accuracy* começam sincronizadas com a *training loss* e *training accuracy*, porém converge bem rápido para uma *accuracy* de 0.89 e logo entra em *overfitting*.





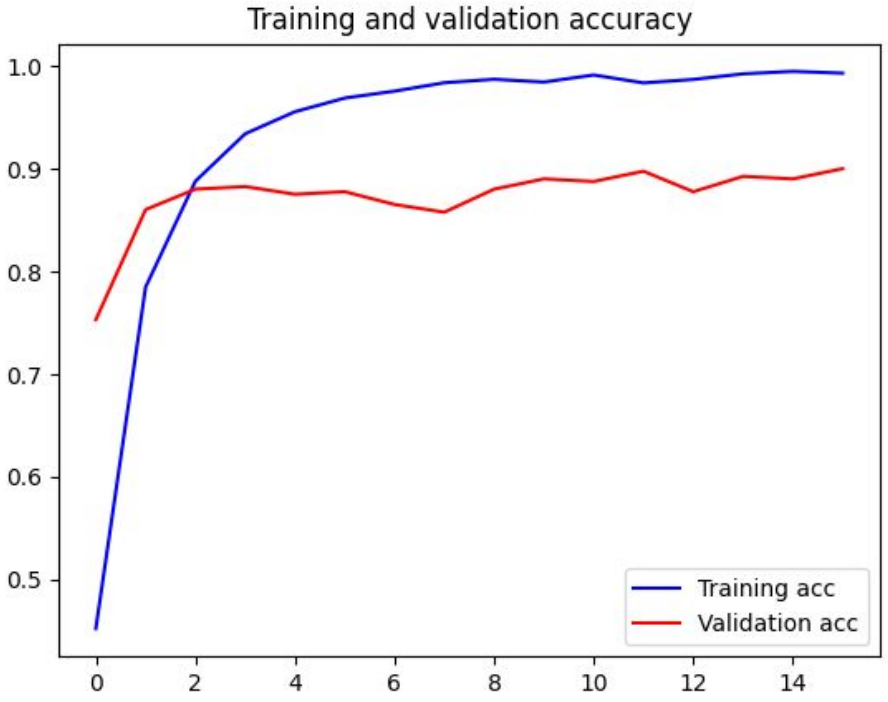

```
[[30  8  0  0  0  0  1  0  0  0  0  0]
 [ 2 26  0  0  0  0  0  0  2  0  1  1]
 [ 1  0 32  0  1  1  0  0  1  0  0  0]
 [ 0  0  0 31  3  0  0  0  1  3  1  0]
 [ 0  1  1  3 30  1  0  0  1  1  0  0]
 [ 2  1  0  1  0 19  4  0  1  0  1  0]
 [ 3  1  0  0  0  0 28  0  0  0  0  0]
 [ 0  0  0  0  1  0  0 22  0  1  1  3]
 [ 0  0  0  0  0  1  0  0 24  3  2  1]
 [ 0  0  0  0  0  0  0  0  2 27  0  1]
 [ 0  0  0  0  0  0  0  0  0  0 34  0]
 [ 0  1  0  0  0  1  0  0  6  0  4 21]]
```

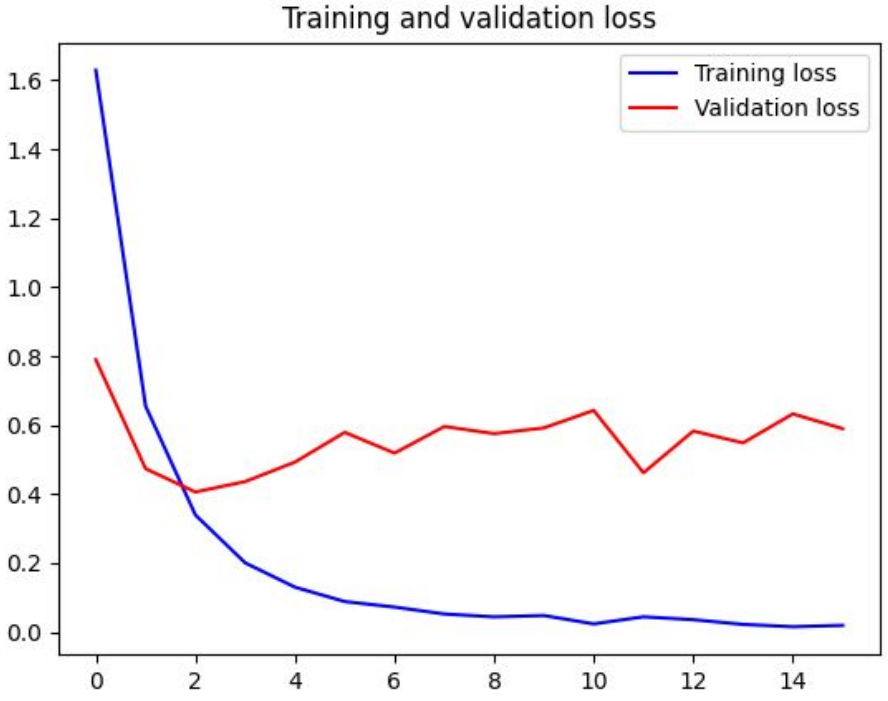
A matriz de confusão para esse modelo apesar de como na CNN apresentar as mesmas confusões, o desempenho geral em relação às confusões foi maior.

3.2.1 Com Data Augmentation, epochs = 16, learning_rate = 0.01 , batch = 64.

Para melhor visualização dos dados no momento em que entra em overfitting, segue a seguir a representação para 16 épocas, é possível visualizar

com mais precisão que o desempenho máximo chega em 0.89 com duas épocas e entra em overfitting, esse modelo convergiu bem rápido.



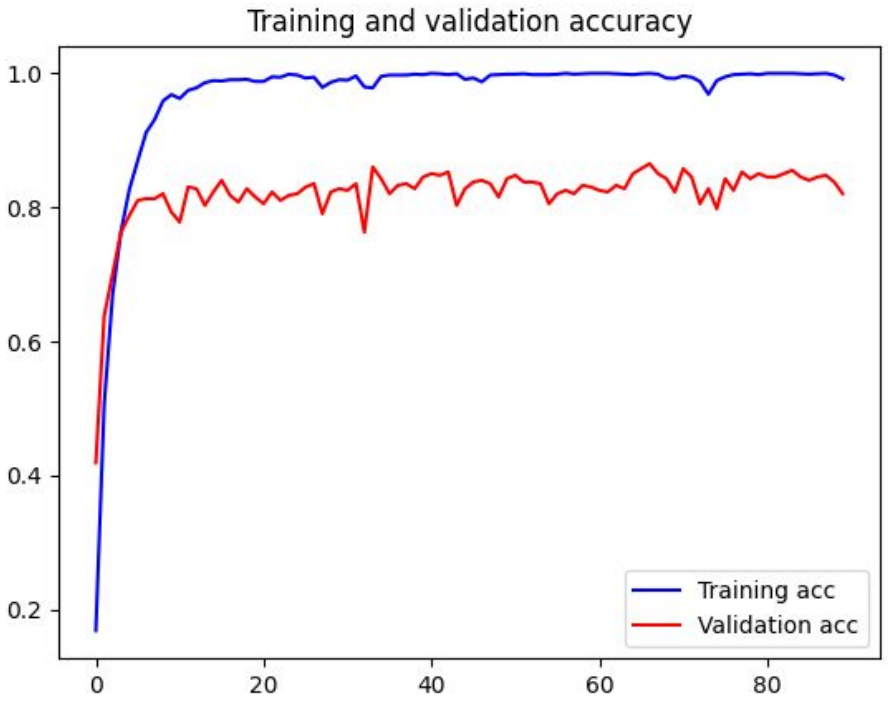


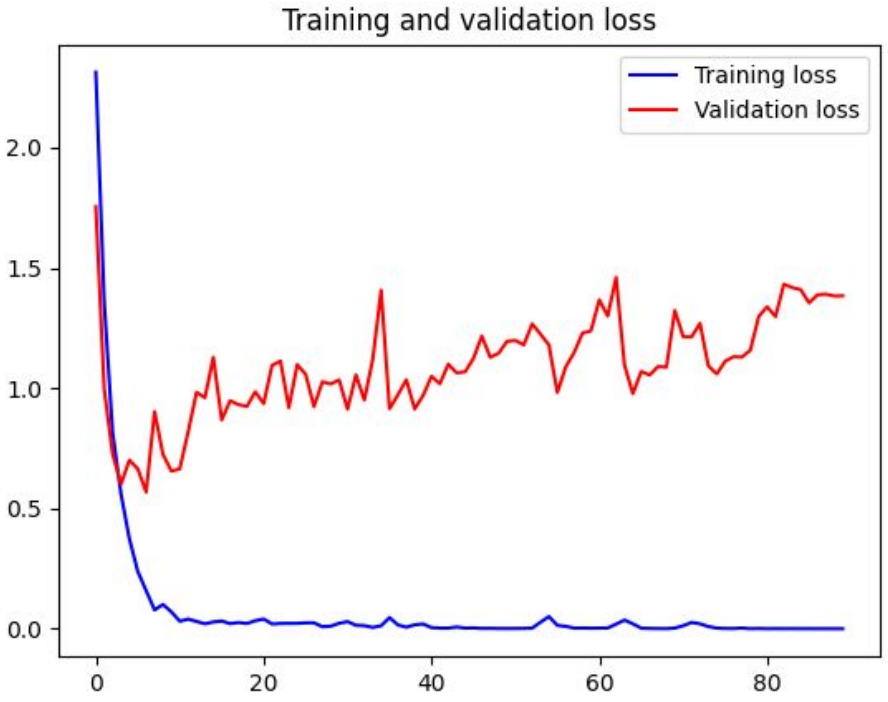
```
[[36  1  0  0  0  1  1  0  0  0  0  0]
 [ 2 28  0  0  0  0  0  0  1  0  1  0]
 [ 0  0 35  0  1  0  0  0  0  0  0  0]
 [ 1  0  0 33  1  0  1  3  0  0  0  0]
 [ 1  1  0  3 32  0  0  0  0  1  0  0]
 [ 1  0  0  0  0 26  1  1  0  0  0  0]
 [ 1  0  0  0  0  0 31  0  0  0  0  0]
 [ 0  0  0  0  0  1  0 27  0  0  0  0]
 [ 1  0  0  1  0  0  0  0 26  1  1  1]
 [ 0  0  0  0  0  0  0  0  1 28  0  1]
 [ 0  0  0  0  0  0  0  0  0  0 33  1]
 [ 1  0  0  0  0  2  0  1  0  0  3 26]]
```

A matriz de confusão para o LeNet 5 com 16 épocas se mostra ainda melhor em desempenhos gerais que todas as outras ja apresentadas, em geral apresentando ainda as mesmas confusões, porém bem menos frequentes.

3.2.1 Sem Data Augmentation, epochs = 90, learning_rate = 0.01

Esse caso apresentou também um desempenho bem parecido com o modelo que usou Data Augmentation, ele ainda converge bem rápido para uma *accuracy* de 0.89 e logo entra em *overfitting*.





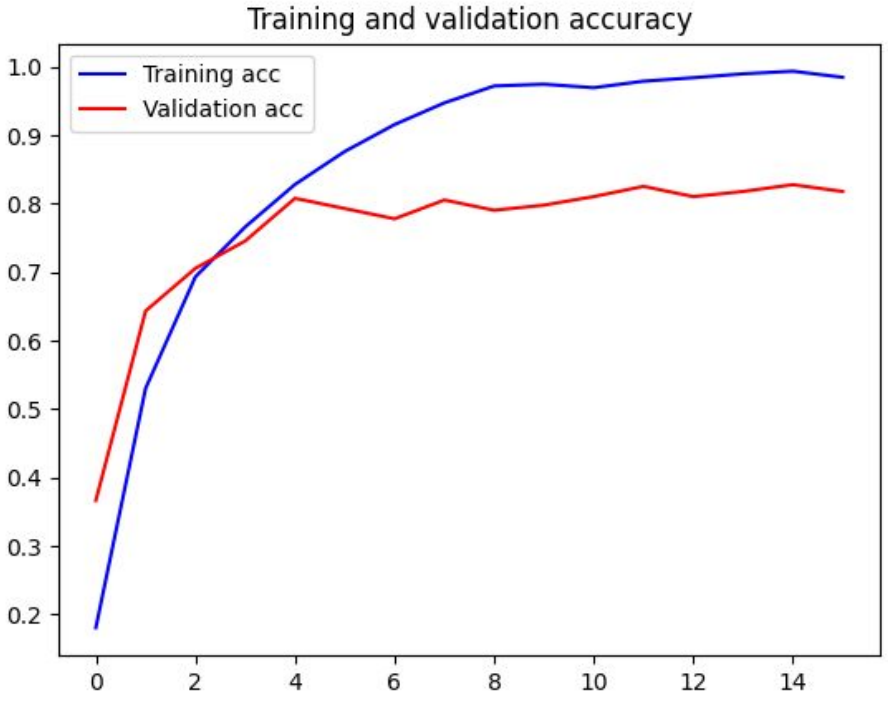

```
[[30  3  0  0  0  1  3  1  0  0  1  0]
 [ 4 26  0  0  0  0  0  1  0  0  1  0]
 [ 0  0 33  0  2  0  0  1  0  0  0  0]
 [ 1  0  1 30  2  0  1  2  1  1  0  0]
 [ 0  2  1  6 25  2  1  0  0  1  0  0]
 [ 2  0  0  0  0 23  4  0  0  0  0  0]
 [ 0  0  0  0  0  2 30  0  0  0  0  0]
 [ 1  0  0  0  0  1  0 24  0  0  0  2]
 [ 1  0  0  1  0  0  0  0 25  2  0  2]
 [ 0  0  0  0  0  0  0  0  0 30  0  0]
 [ 0  2  0  0  0  1  0  0  0  1 30  0]
 [ 0  2  0  0  0  4  0  0  1  0  3 23]]
```

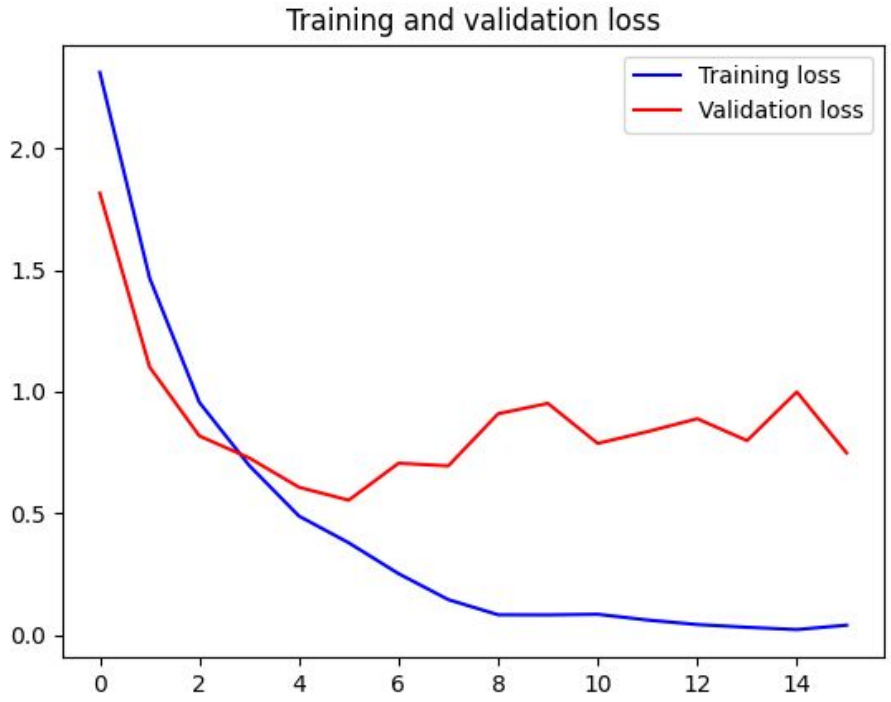
A matriz de confusão apresenta o ja esperado conforme observado em todos os outros anteriormente.

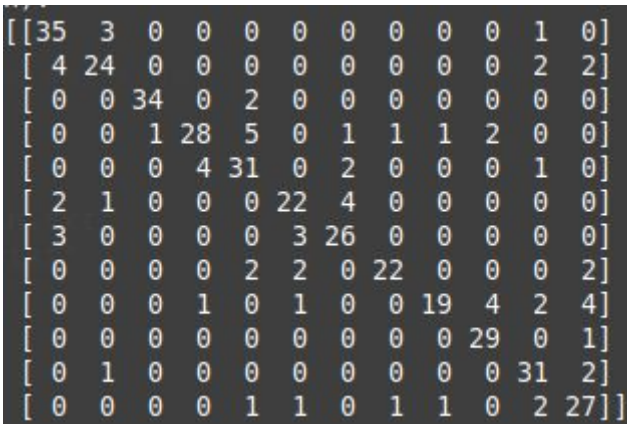
3.2.1 Com Data Augmentation, epochs = 16, learning_rate = 0.01 , batch = 64.

Para melhor visualização dos dados no momento em que entra em overfitting, segue a seguir a representação para 16 épocas, é possível visualizar com mais precisão que o desempenho máximo chega em 0.89 com três épocas e

entra em overfitting, esse modelo convergiu bem rápido, assim como no anterior, porém uma época depois.







```
[[35  3  0  0  0  0  0  0  0  0  1  0]
 [ 4 24  0  0  0  0  0  0  0  0  2  2]
 [ 0  0 34  0  2  0  0  0  0  0  0  0]
 [ 0  0  1 28  5  0  1  1  1  2  0  0]
 [ 0  0  0  4 31  0  2  0  0  0  1  0]
 [ 2  1  0  0  0 22  4  0  0  0  0  0]
 [ 3  0  0  0  0  3 26  0  0  0  0  0]
 [ 0  0  0  0  2  2  0 22  0  0  0  2]
 [ 0  0  0  1  0  1  0  0 19  4  2  4]
 [ 0  0  0  0  0  0  0  0  0 29  0  1]
 [ 0  1  0  0  0  0  0  0  0  0 31  2]
 [ 0  0  0  0  1  1  0  1  1  0  2 27]]
```

Os resultados obtidos com a classificação através da *SVM* foram bastante controversos, sem tempo para refazer e verificar, optei por não apresentar os dados.

4.ImageNet e SVM

4.1Fine-Tuning

4.2Transfer-learning