Cats and Dogs Image Classification

Master in Artificial Intelligence 2023-2024

Professor Vittorio Murino and Professor Andrea Avogaro

Saman Nourani – VR512633

June 2024

Index

Motivation and rationale

It was always a question for me how can we find out what kind of animal this is with neural network and code, so it was important for me to be able to answer this question for myself. It was my first complete and comprehensive translation for the use of artificial intelligence, algorithms, and models, and I learned many important and interesting points in this project. I used the ResNet-50 and CNN models, and I had a hard time with the ResNet-50 model. Because it takes a long time to process. I tried a lot to increase the accuracy of this model, so I was able to increase the accuracy of the diagnosis according to what I wanted.

Objectives

As I said, the question for me was how to identify animals with high accuracy. I chose two animals that we can see more than others among people; dogs and cats Of course, distinguishing these two animals was more interesting for me, even though I don’t have a good relationship with cats, unlike dogs. Further, I came to the conclusion that with this project I can reach the questions that are in my mind about animal recognition. I got acquainted with different algorithms, finally, I chose the ResNet and CNN algorithm.

Methodology

- **Methods and Algorithms:**
  - **Neural Network:** I used the neural network for vision. Pixel by pixel to better recognize photos of dogs and cats.
  - **ResNet:** I used the reset to know the environment and have a solution for diagnosis.

- **Data set:**

  I used the Kaggle dataset, which includes hundreds of photos of dogs and cats.

- **Analytical tools:**
  - **Tensorflow and Keras:** I used TensorFlow, which is one of the most famous and popular open sources of machine learning. I used Cross for API to train deep learning models.
  - **Image data generator:** I used Bay to better recognize and evaluate the photo.
  -

- **Computational Tools:**
  - **python:** I used Python, which is the most popular and well-known programming language all over the world these days. It is also very popular in the field of computer and machine learning.
  - **Matplotlib:** I used Matplotlib for trading and visualization.

Experiments and Results

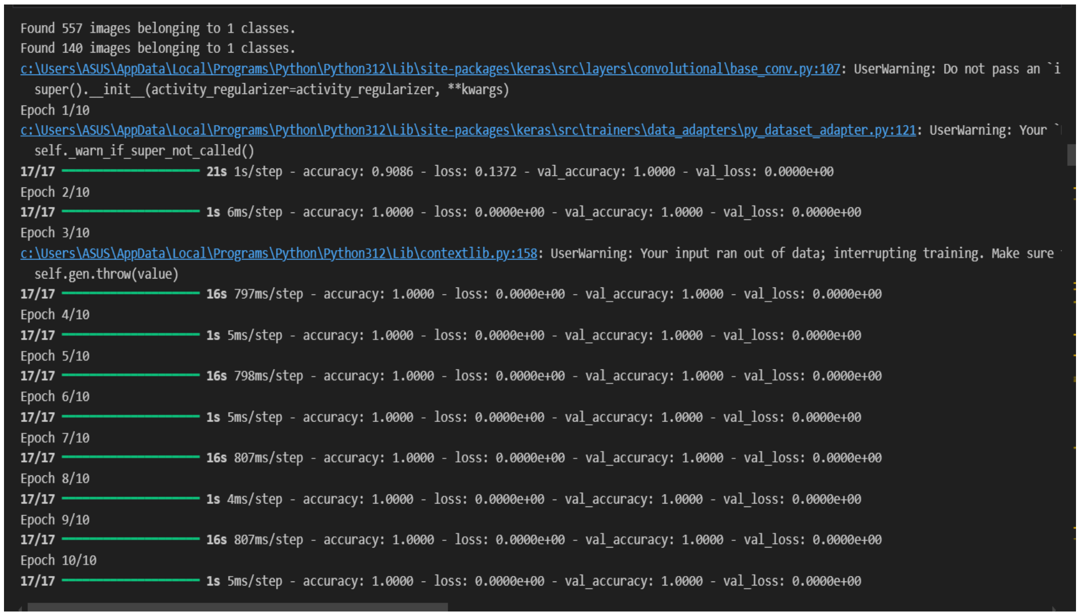
- **Evaluation Protocol:**
  - **Model Training:** I trained different models using ResNet and CNN. I updated the data and photos of dogs and cats that I got from Kegel.
  - **Validation:** During training, I validate the models using a separate validation set, which is typically a subset of the training data that the model does not see during training. This helps to monitor model performance and detect overfitting and better fit.
  - **Testing:** After training and processing, tests and tests are conducted on the test data, which is a subset of the data set and the original photos of dogs and cats taken from Kegel. The positive point of this work is that it can show us the small points that have not been paid attention to in the project. In this case, we can see which part did not work well or correctly**.**

- **Conditions**
  - **Hardware:** I had a good system/ laptop for running the deep learning's code, such as: available and suitable GPU, good speed for running, CPU to control the RAM.
  - **Software:** I just need an environment to run the Python code, such as VS code, Google Colab, PyCharm. And some necessaries libraries.
  - **Data Processing:** I just need a standard techniques for image processing, pixel by pixel (to 224*224 pixel) and scaling pixel by [0, 1].

- **Confusion Matrix**
  - **Accuracy:** The simplest, easiest, and one of the best ways for evaluate classification performance is accuracy. Which can measure the accuracy of thousands of photos and show us.

    Here some of the result accuracy that I check / code:

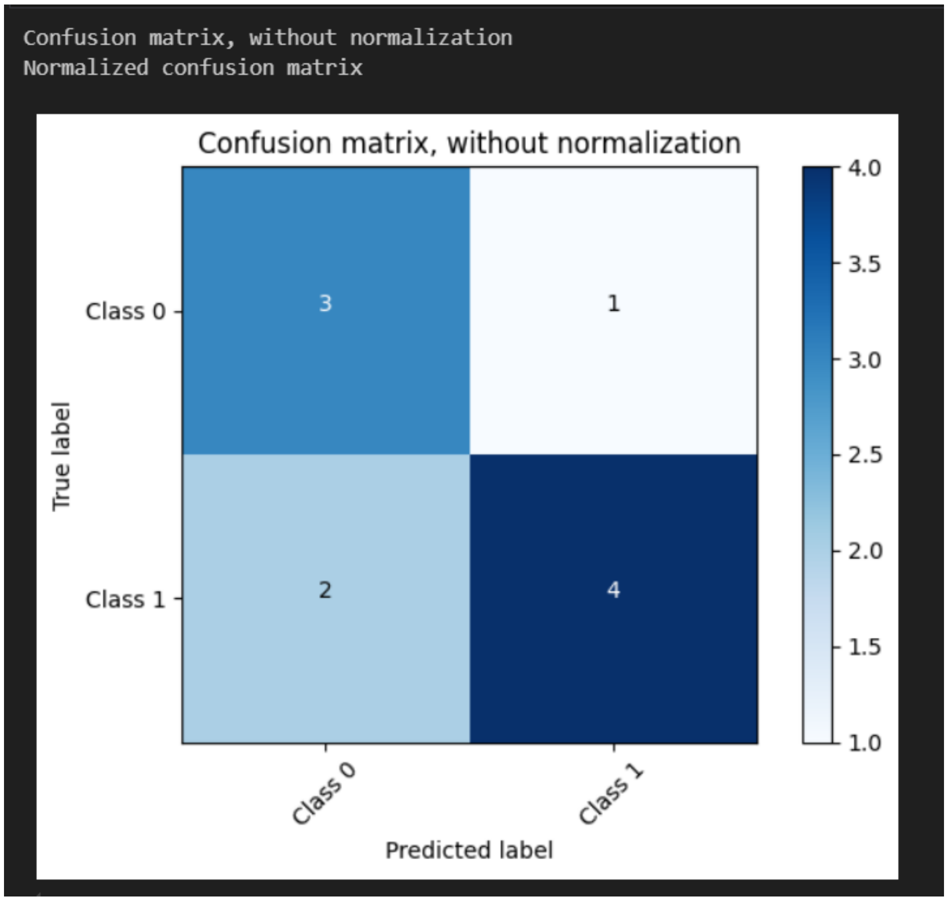
    I checked the accuracy with 10 Epoch, In this picture we can see the result of 10 different accuracy. There are 0.99 / about 1.00



```
Found 557 images belonging to 1 classes.
Found 140 images belonging to 1 classes.
c:\Users\ASUS\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `i
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
c:\Users\ASUS\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `
  self._warn_if_super_not_called()
17/17 ━━━━━━━━━━━━━━━━━━━━ 21s 1s/step - accuracy: 0.9086 - loss: 0.1372 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 2/10
17/17 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 3/10
c:\Users\ASUS\AppData\Local\Programs\Python\Python312\Lib\contextlib.py:158: UserWarning: Your input ran out of data; interrupting training. Make sure
  self.gen.throw(value)
17/17 ━━━━━━━━━━━━━━━━━━━━ 16s 797ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 4/10
17/17 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 5/10
17/17 ━━━━━━━━━━━━━━━━━━━━ 16s 798ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 6/10
17/17 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 7/10
17/17 ━━━━━━━━━━━━━━━━━━━━ 16s 807ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 8/10
17/17 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 9/10
17/17 ━━━━━━━━━━━━━━━━━━━━ 16s 807ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 10/10
17/17 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
```

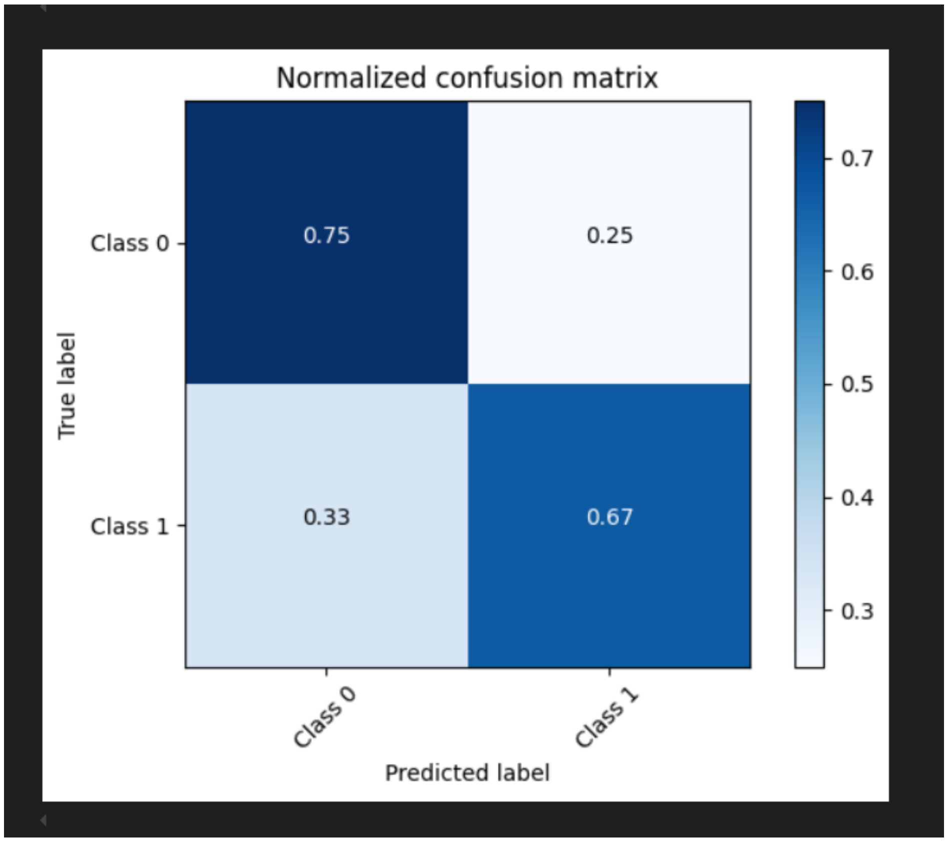
- **Confusion matrix:** Confusion matrix provides accurate separation of predictions versus actual data. It shows the number of true positives, false positives, true negatives and false negatives. It helps to understand where the model has errors and where it works correctly.

  The first one is a simple one, without normalization.



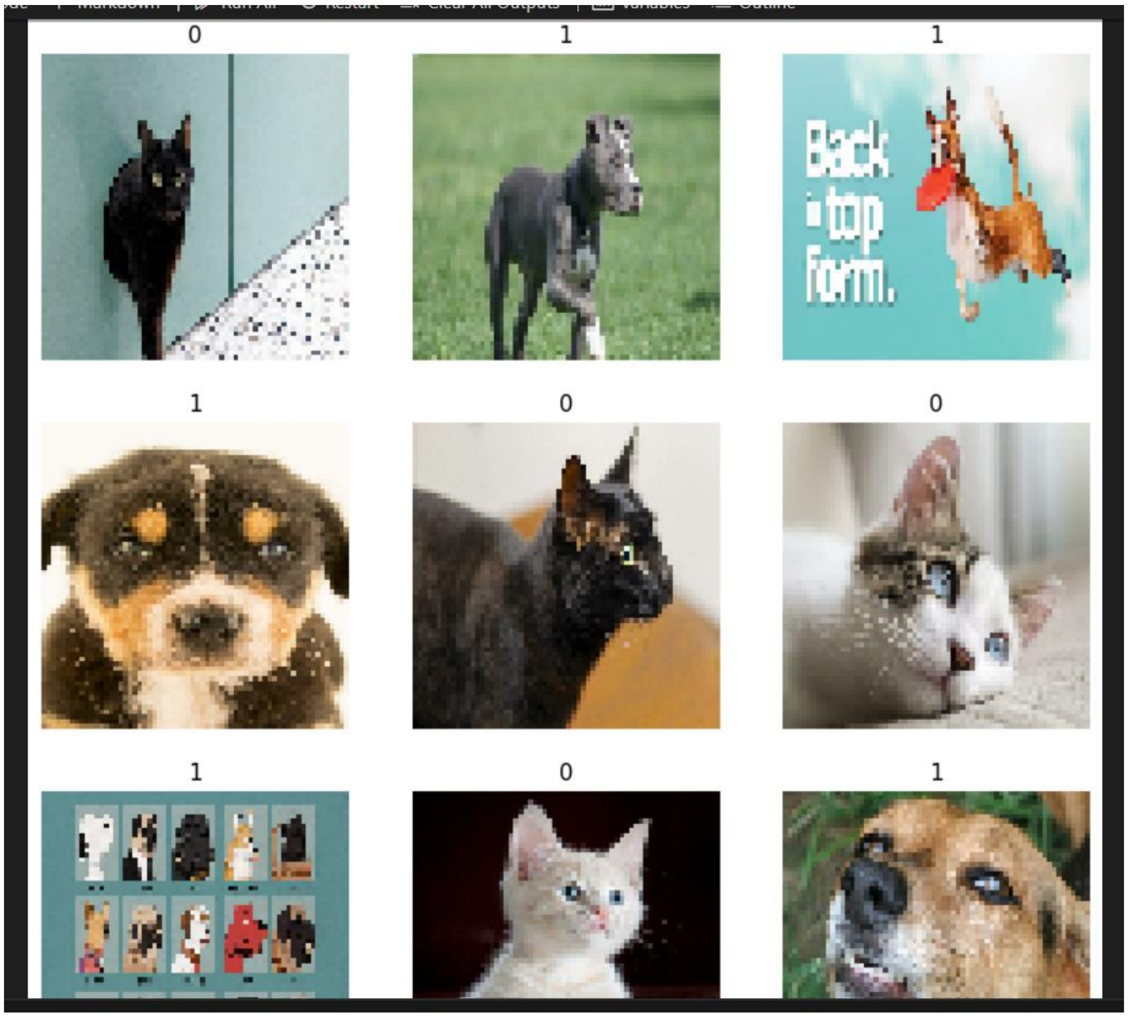
```
Confusion matrix, without normalization
Normalized confusion matrix
```

The second one is Normalized confusion matrix.



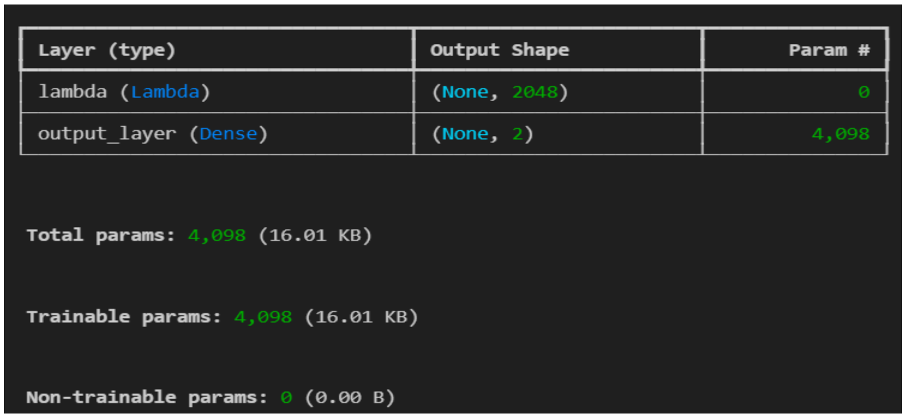
Goals and results

These models and algorithms show us which one is dog amd which one is cat. I code like if an image is a dog show the number 1, and if an image is a cat, show the number 0.



Model Structure

- **Model Type:** Sequential: It means the model consists of layers that are stacked sequentially.
- **Layers:**
  - Lambda Layer:
    - shape (None, 2048), The output from this layer has a shape of (batch size, 2048), where None represents the batch size which can vary.
    - Parameters: 0, This layer does not have any trainable parameters. It typically applies a custom function or operation to the input data.
  - Dense Layer:
    - shape (None, 2), The output from this layer has a shape of (batch size, 2), indicating that there are 2 units (neurons) in this layer, which matches the number of classes in your classification problem.
    - Parameters: 4098, This layer has 4098 trainable parameters such as weights and biases.



```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━┓
┃ Layer (type)                  ┃ Output Shape         ┃      Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━┩
│ lambda (Lambda)               │ (None, 2048)         │            0 │
├───────────────────────────────┼──────────────────────┼──────────────┤
│ output_layer (Dense)          │ (None, 2)            │        4,098 │
└───────────────────────────────┴──────────────────────┴──────────────┘

 Total params: 4,098 (16.01 KB)

 Trainable params: 4,098 (16.01 KB)

 Non-trainable params: 0 (0.00 B)
```

Conclusion

This project aimed to classify images of cats and dogs using deep learning techniques. The dataset was split into training and test sets, and images were preprocessed with normalization and augmentation to improve model performance. Two model architectures were used: a custom convolutional neural network (CNN) and a transfer learning approach with ResNet-50. The custom CNN included convolutional, pooling, and dense layers, while the ResNet-50 model utilized pretrained weights for improved accuracy.

The models were trained with callbacks for early stopping and learning rate reduction to prevent overfitting. Evaluation metrics such as accuracy and confusion matrix were used to measure performance. The custom CNN achieved reasonable accuracy, while the transfer learning model with ResNet-50 showed superior performance due to the robust features learned from a large dataset.

In conclusion, the project demonstrated the effectiveness of both custom CNNs and transfer learning models for image classification. Data augmentation and regularization techniques were crucial for model generalization. The comprehensive evaluation using multiple metrics provided insights into the models' strengths and areas for improvement, suggesting that fine tuning pre-trained models and exploring advanced augmentation techniques could further enhance performance.

References

1- Kaggle ResNet-50 - https://www.kaggle.com/code/adityapramar15/resnet50-catsdogs