Image Recognition

NEURAL NETWORKS, BINARY CLASSIFICATION

MACHINE LEARNING

Muhammed Besim Sakaoglu

985861

April 2023

ABSTRACT

The aim of this project is to develop a CNN to perform binary classification on the given cats and dogs' images dataset. I explored different network architectures, effects of different hyperparameters and some regularization techniques. Finally, the ultimate model has developed through 5fold Cross-Validation.

Introduction

Convolutional neural nets are phenomenally effective models for performing image recognition tasks today. In this study, I constructed a total of 4 Convolutional Neural Networks for classification of cats and dogs using a famous dataset, examining the impact of different network structures and techniques on the performance of the models. Foremost, I started with a basic model namely Model 1, and I tried to experiment with some hyperparameter tuning and regularization actions in order to overcome *Overfitting* problem. Ultimately, I have reached a stand point for my 5-fold CV model.

Image classification on the other hand, has many applications in computer vision, such as face recognition, medical diagnosis, and self-driving cars. For this project, we were responsible for the collections of images that have been labeled by human annotators and are widely used by researchers and practitioners. One such dataset is the Dataset, which consists of around 25000 photos of cats and dogs. The Dataset is a well-known dataset in the field of computer vision, as it sprang to prominence after winning a Kaggle competition in 2013.

The Dataset is not only popular but also useful for image classification. It provides a balanced and realistic set of images that capture the diversity and difficulty of natural images.

My task was literally this:

'Use Tensorflow 2 to train a neural network for the binary classification of cats and dogs based on images from this dataset. Images must be transformed from JPG to RGB (or grayscale) pixel values and scaled down. Experiment with different network architectures and training parameters documenting their influence of the final predictive performance. Use 5-fold cross validation to compute your risk estimates. While the training loss can be chosen freely, the reported cross-validated estimates must be computed according to the zero-one loss.'

So, I applied the processes below accordingly.

- ***Removing*** damaged photos: Some photos in the original dataset were **corrupted or unreadable**. These photos were removed to avoid errors and inconsistencies.

- Converting original.Jpg images to RGB: The original images were stored in JPEG format, which uses a lossy compression algorithm that can degrade the image quality. To preserve the image information and avoid artifacts, the images were converted ***to RGB format***, which uses three channels (red, green, and blue) to represent the color of each pixel.

- ***Scaling*** to size of 150 height, 150 width: The original images had different sizes and aspect ratios, which can affect the model performance and efficiency. To make the images uniform and compatible with the model input, they were scaled to a standard size of 150 pixels in height and 150 pixels in width.

- Normalizing RGB values from [0,255] to [0,1] for optimal model performance: The RGB values of each pixel range from 0 to 255, which can cause numerical instability and gradient explosion in the model training. To avoid these problems and improve the model convergence, the **RGB values were normalized from [0,255] to [0,1] by dividing them by 255**.

By applying these preprocessing steps, the Dataset becomes a suitable benchmark for image classification. It allowed me to compare different models and techniques on a common ground and measure their accuracy and efficiency.

Key Concepts to remember:

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is particularly well-suited for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers [1]. Some key concepts and definitions in CNN machine learning are:

Filters: Determines the depth of the output feature map.

Batch size: In the context of machine learning, batch size refers to the quantity of training examples used in a single iteration.[2]

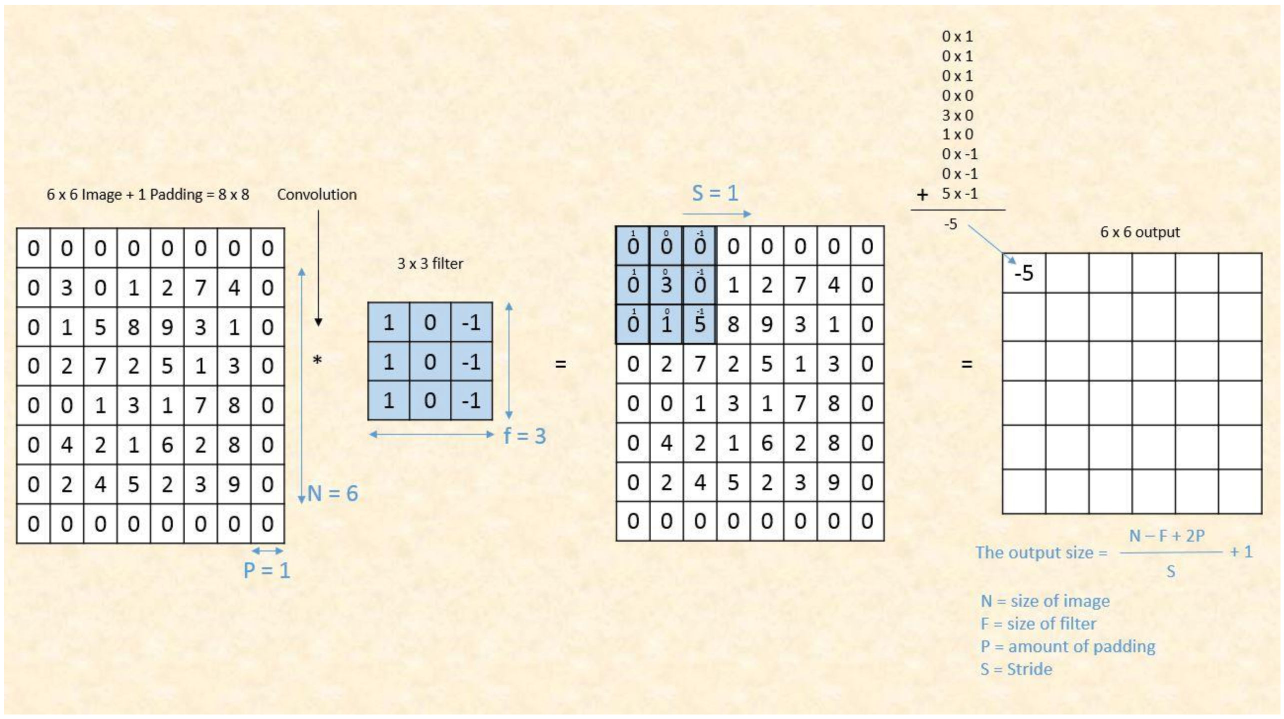
Kernel: Also referred to as the size of the filter.

Padding: Solves the shrinkage of the kernel output feature map by adding a given amount of pixels.

Stride: Influences the size of the output map by determining the distance between two tiles where the convolution is performed.

Activation Function: Defines whether a neuron is activated or not.

[1] https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/
[2] https://radiopaedia.org/articles/batch-size-machine-learning?lang=us



Figure[3] Convolution on 6 x 6 image with zero padding = 1

Model 1- Basic Model

Model 1 is a sequential model.

This model is quite straight forward and composed of:

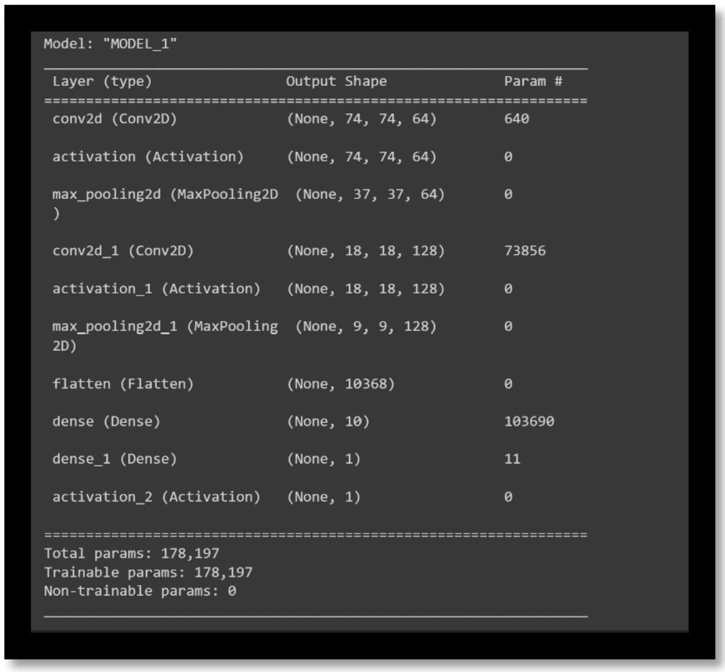
In the convolutional layer, the input is rescaled by dividing it by 255 and the shape of the input is specified as a grayscale image with dimensions of 150x150. (150,150,1)

[3] https://gaussian37.github.io/dl-concept-cnn/

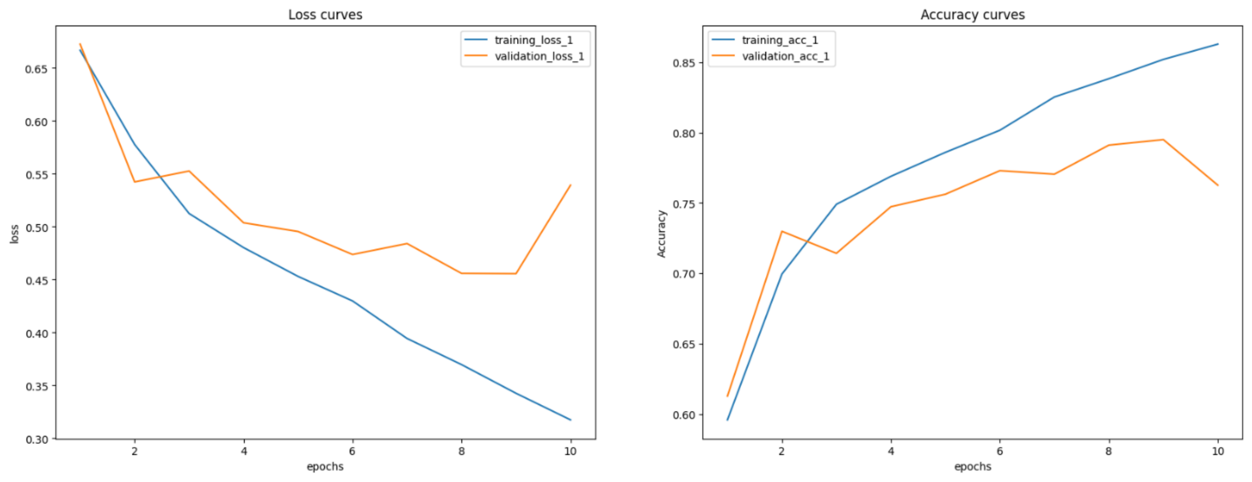
```
Text(0.5, 0, 'shape: (150, 150)')
```

- Has **Adam optimizer** and 178,197 trainable parameters.(0 non-trainable)
- The validation accuracy was around 76%.
- Stride was 2.

The first layer is a 2D convolutional layer with 64 filters and a kernel size of (3,3) followed by a **ReLU activation function** and a max pooling layer with pool size (2,2). The second convolutional layer has 128 filters and a kernel size of (3,3) followed by another ReLU activation function and max pooling layer with pool size (2,2). The output is then flattened and passed through a **dense layer** with **10** units. The **final dense layer** has **1** unit with a **sigmoid activation function** since it is binary classification of cats or dogs images.



```
Model: "MODEL_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 74, 74, 64)        640

 activation (Activation)     (None, 74, 74, 64)        0

 max_pooling2d (MaxPooling2D  (None, 37, 37, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 18, 18, 128)       73856

 activation_1 (Activation)   (None, 18, 18, 128)       0

 max_pooling2d_1 (MaxPooling  (None, 9, 9, 128)        0
 2D)

 flatten (Flatten)           (None, 10368)             0

 dense (Dense)               (None, 10)                103690

 dense_1 (Dense)             (None, 1)                 11

 activation_2 (Activation)   (None, 1)                 0

=================================================================
Total params: 178,197
Trainable params: 178,197
Non-trainable params: 0
_________________________________________________________________
```



Over-fitting outcome was visible with accuracy and loss curves even though it has not reached a plateau!

Model 2 (Hyperparameter Tuning)

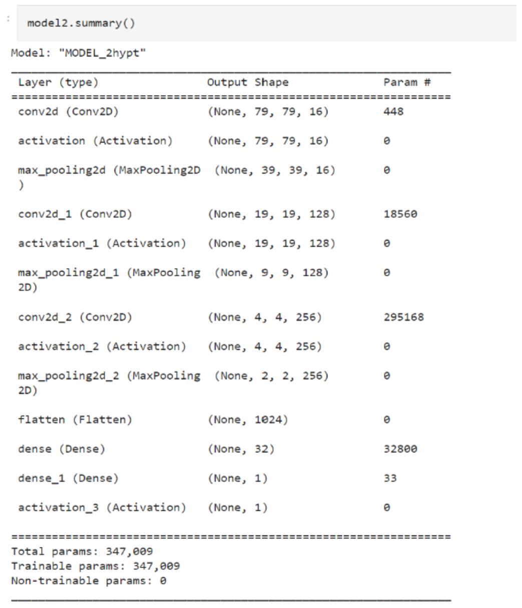
The second model named “MODEL_2hypt” that consists of several layers. The first layer is a 2D convolutional layer with 16 filters and a kernel size of (3,3) followed by a ReLU activation function and a max pooling layer with pool size (2,2). The second convolutional layer has 128 filters and a kernel size of (3,3) followed by another ReLU activation function and max pooling layer with pool size (2,2). The third convolutional layer has 256 filters and a kernel size of (3,3) followed by another ReLU activation function and max pooling layer with pool size (2,2). The output is then flattened and passed through a dense layer with 32 units. The final dense layer has 1 unit with a sigmoid activation function.

`Key changes compared to previous model:`

- Image sized increased to 160x160, it was 150x150 before.
- Batch size decreased to 90, it was 100 before.
- Epochs increased to 20, it was 10 before. (it has not reached to plateau!)
- Added hidden layer for model2hypt and made it deeper.

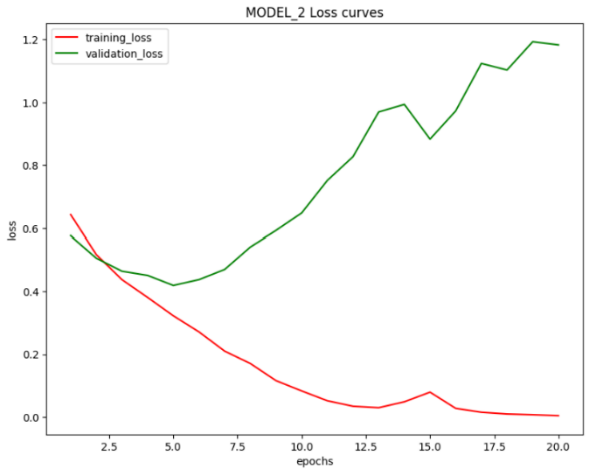
• Added padding aiming to result in a more compact representation of the input, but I avoided computational costs by choosing it is `valid` unlike `same`. There were no padding before.

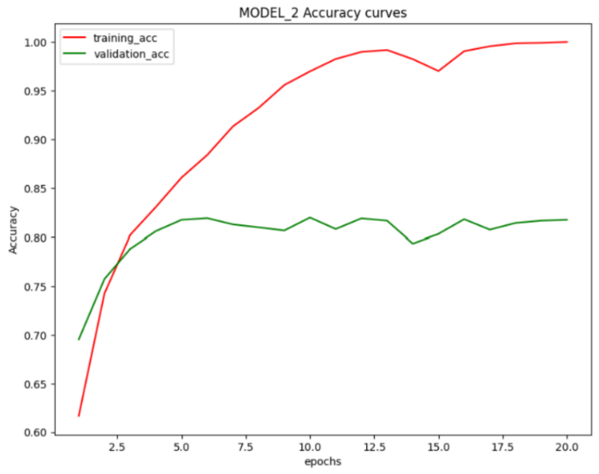
Accuracy: `validation accuracy: 0.8174` has increased compared to basic model.



```
model2.summary()
```

```
Model: "MODEL_2hypt"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 79, 79, 16)        448

 activation (Activation)     (None, 79, 79, 16)        0

 max_pooling2d (MaxPooling2D  (None, 39, 39, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 19, 19, 128)       18560

 activation_1 (Activation)   (None, 19, 19, 128)       0

 max_pooling2d_1 (MaxPooling  (None, 9, 9, 128)        0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 256)         295168

 activation_2 (Activation)   (None, 4, 4, 256)         0

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 256)        0
 2D)

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 32)                32800

 dense_1 (Dense)             (None, 1)                 33

 activation_3 (Activation)   (None, 1)                 0

=================================================================
Total params: 347,009
Trainable params: 347,009
Non-trainable params: 0
_________________________________________________________________
```



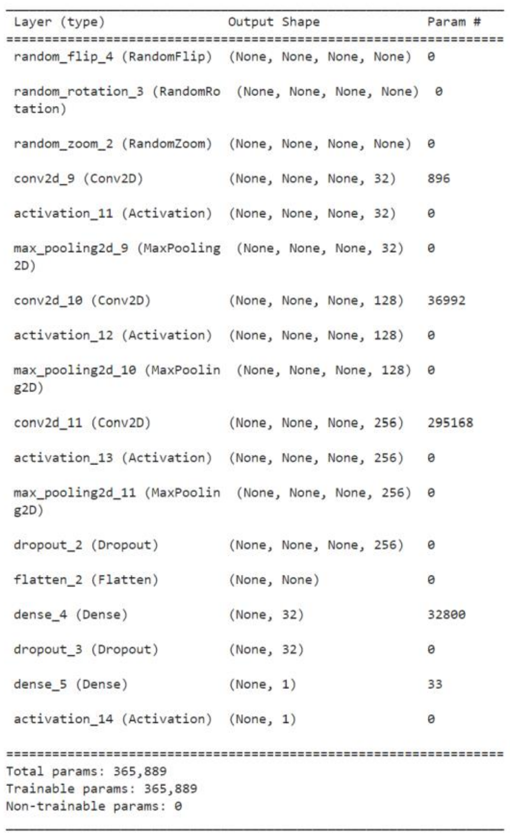


We achieved higher accuracy without adding too much complexity. For instance, trainable parameters are still around the previous one and not even 7 digits. However, we still are facing the over-fitting issue even though we almost handled the reaching plateau part.

Model 3 (Regularization)

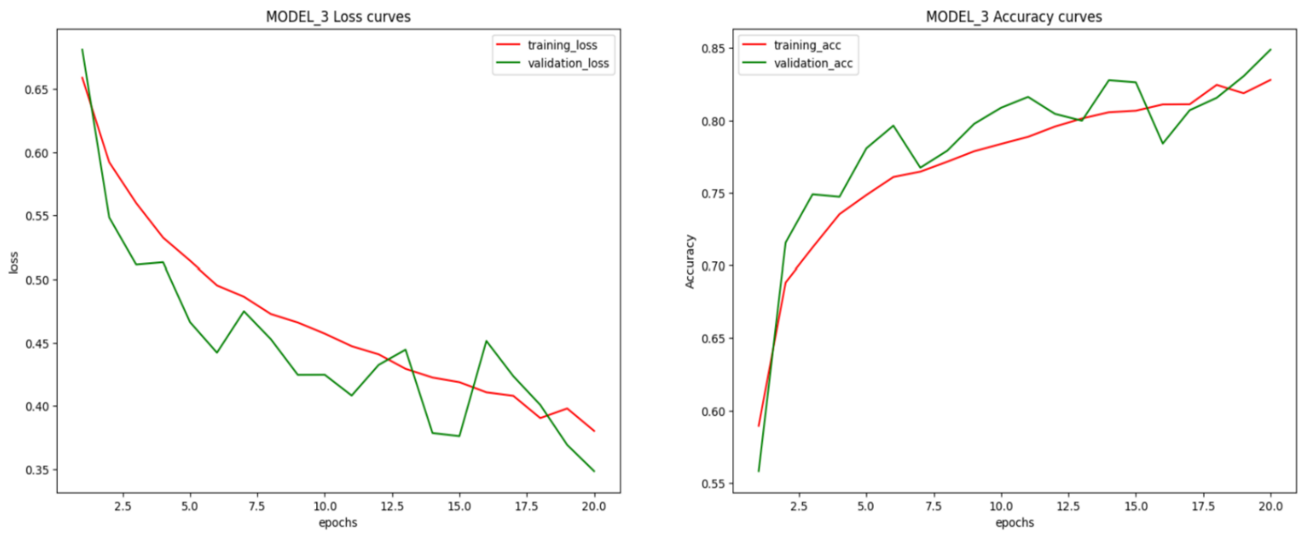
Model3 is Data Augmentation and Dropout included version of Model 3 and deals with Overfitting that we faced earlier. Another saying is that Model 3 is an improved version of Model2hypt.

The first three layers perform data augmentation by randomly flipping the input horizontally, **rotating** it by up to 15%, and **zooming** it by up to 15%. The next layer is a 2D convolutional layer with 32 filters and a kernel size of (3,3) followed by a ReLU activation function and a max pooling layer with pool size (2,2). The second convolutional layer has 128 filters and a kernel size of (3,3) followed by another ReLU activation function and max pooling layer with pool size (2,2). The third convolutional layer has 256 filters and a kernel size of (3,3) followed by another ReLU activation function and max pooling layer with pool size (2,2). A **dropout layer** with a rate of %25 is then applied. The output is then flattened and passed through a dense layer with 32 units followed by **another dropout layer** with a rate of %30. The final dense layer has 1 unit with a sigmoid activation function.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 random_flip_4 (RandomFlip)  (None, None, None, None)  0

 random_rotation_3 (RandomRo  (None, None, None, None)  0
 tation)

 random_zoom_2 (RandomZoom)  (None, None, None, None)  0

 conv2d_9 (Conv2D)           (None, None, None, 32)    896

 activation_11 (Activation)  (None, None, None, 32)    0

 max_pooling2d_9 (MaxPooling  (None, None, None, 32)   0
 2D)

 conv2d_10 (Conv2D)          (None, None, None, 128)   36992

 activation_12 (Activation)  (None, None, None, 128)   0

 max_pooling2d_10 (MaxPoolin  (None, None, None, 128)  0
 g2D)

 conv2d_11 (Conv2D)          (None, None, None, 256)   295168

 activation_13 (Activation)  (None, None, None, 256)   0

 max_pooling2d_11 (MaxPoolin  (None, None, None, 256)  0
 g2D)

 dropout_2 (Dropout)         (None, None, None, 256)   0

 flatten_2 (Flatten)         (None, None)              0

 dense_4 (Dense)             (None, 32)                32800

 dropout_3 (Dropout)         (None, 32)                0

 dense_5 (Dense)             (None, 1)                 33

 activation_14 (Activation)  (None, 1)                 0

=================================================================
Total params: 365,889
Trainable params: 365,889
Non-trainable params: 0
_________________________________________________________________
```

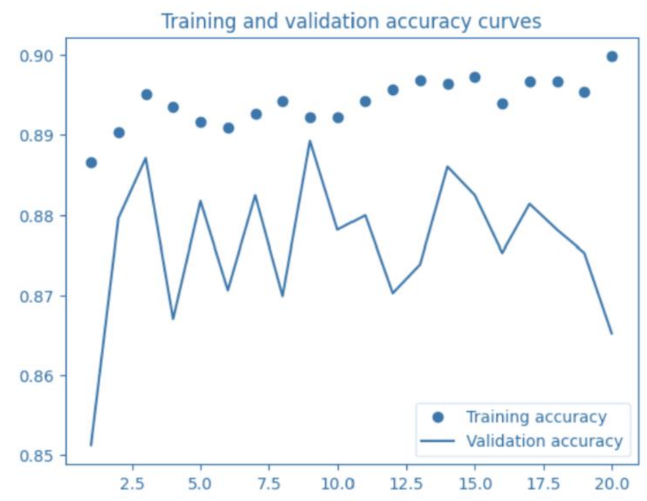
```
Test Loss: 0.3484468460083008
Test Accuracy: 0.8489961624145508
```



Finally, we got higher accuracy rates and lower loss rates without adding too much complexity and trainable parameters were almost the same as the second model. Also, over-fitting is not an issue anymore and we can continue with this model in order to investigate 5fold CV. (Note: Some spikes that validation accuracy/loss curves cross training curves can be ignored. It happened because of additional dropout layer.)

Model CV (5-fold)

This is the final model that I investigated, and it is considered the best performing model so far, as with cross-validation it repeats the procedure of fitting and evaluating the NN for each iteration. To illustrate, when cross-validation is applied to a dataset, all of the data is used to make predictions. The dataset is split into k-folds (5 this time), and the model is trained and verified over 5 iterations. The model is trained on four of the folds and verified on the remaining fold in each iteration. With each iteration, the training and validation folds change.



```
val_accuracy did not improve from 0.88925
```

Conclusion

In this study, I gained a deep understanding of how neural networks work and how using a convolutional neural network (CNN) is more effective. I also observed different CNN architectures and learned how to address overfitting by introducing regularization methods like adding dropout layer and data augmentation and tuning hyperparameters such as batch size and epochs. I also discovered that adding more convolutional layers, without making it so complicated, can improve the model's performance. Since I have an old laptop that does not have TPU, also restricted GPU (MacOS problem), allowed me to achieve this accuracy rate. I could have discovered higher accuracies with higher specs on my laptop or subscribed cloud plan like Google`s Colab Premium.