UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY
CLASS 19CLC5

ARTIFICIAL INTELLIGENCE
PROJECT 03

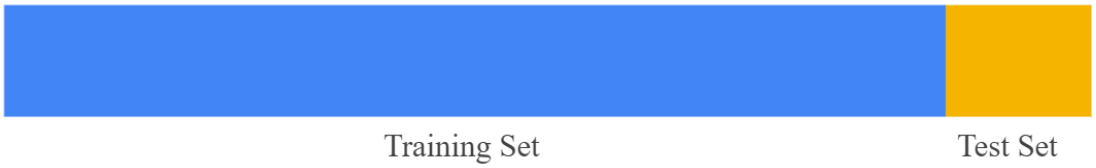
Group member

19127268 Nguyễn Ngọc Thanh Tâm
19127511 La Ngọc Hồng Phúc
19127568 Nguyễn Thị Minh Thu

Contents

1. Training set, validation set, test set

We usually divide the data set into 2 subsets: training set and test set. Training set is a subset used to train the model, to fit the parameters of the classifier. The model evaluates the training data repeatedly to learn more about the data's behavior. Test set is a subset used to test the trained model. Test data provides a final, real-world check of an unseen dataset to confirm that the ML algorithm was trained effectively.



Traditionally, the test set is locked away completely until all model tuning is complete. You must not use the test data to train the model. If after training the model and testing it using the test set, you receive a high accuracy, this might indicate that the test data has leaked into the training set. Because you have trained on some of our test data, and as a result, you're no longer accurately measuring how well our model generalizes to new data. If the trained model does about as well on the test data as it does on the training data. We can say that this model does not overfit the training data. We will investigate more about overfitting and underfitting in the below section.

To reduce the chance of overfitting, we can partition the data set into 3 subsets shown in the following figure

What is a validation set? We use the validation set to evaluate results from the training set. Then, use the test set to double-check our evaluation after the model has "passed" the validation set. Validation data provides the first test against unseen data, allowing us to evaluate how well the model makes predictions based on the new data.

The difference between training set and test set is clear: one trains a model, the other confirms that it works correctly. How about the difference between training set and validation set? Validation data provides an initial check that the model can return useful predictions in a read-world setting, which training data cannot do.

There are some confusions about the difference between test set and validation set. In general, both test set and validation set have the same purpose: to qualify the performance, but the main difference is validation set helps us tune the parameters of a classifier, while test set is used only to assess the performance of a fully specified classifier. If data scientists include a separate stage for validation data analysis, the validation data is typically labeled so that they can collect metrics that they can use to better train the model. In this case, the validation set occurs as part of the model training process.

2. Overfitting and underfitting

a. Overfitting

Definition: Overfitting happens when a statistical model fits exactly against its training dataset or corresponds too closely to a particular set of data, which causes inaccurate performance of the algorithm against unseen or new data.

Detection: The data is separated into two main parts: a test set and a training set. By splitting the dataset, we can examine the performance of the model on each set of data to detect overfitting when it occurs. The performance can be measured using the percentage of accuracy observed in both datasets to conclude on the presence of overfitting. If the model performs better on the training set than on the test set, it means that the model is likely overfitting.

Prevention:

- Training with more data: This way of preventing overfitting makes it easy for algorithms to detect the signal better and minimize errors, hence increase the accuracy of the model. However, the additional data the users input must be clean and relevant or else we are just adding more noise and complexity to the model, causing overfitting.
- Feature selection: This is the process of identifying the most important features among several parameters of features within the training dataset and then eliminating the irrelevant or redundant ones. The method helps to simplify the model to establish the dominant trend in the data.
- Cross-validation: Using k-fold cross-validation to assess accuracy of the model. In k-fold cross-validation, the dataset is split into k equally sized subsets, which are also called "folds". One of the k-folds will act as the test set and the remaining folds will be the training set. This process repeats until each of the fold has acted as a test set. After each evaluation, a score is retained and when all iterations have completed, the scores are averaged to assess the performance of the overall model.
- Data augmentation: A less expensive alternative to training with more data. Data augmentation makes a sample data look slightly different every time it is processed by the model and prevents the model from learning the characteristics of the datasets. Another option is adding some noise to the input and output data. Adding noise to the input makes the model more stable without affecting data quality and privacy, while adding noise to the output makes the data more diverse. However, noise addition should be done with moderation so that the extent of the noise is not so much as to make the data incorrect or too different.
- Early stopping: This method seeks to pause the training process before the model starts learning the noise within the model. This approach risks halting training too soon, leading to the opposite problem of underfitting. Finding the "sweet spot" between underfitting and overfitting is the goal here.
- Ensemble methods: These are machine learning methods that combines predictions from multiple separate models. The most popular ensemble methods are boosting and bagging. Boosting works by using simple base models to increase their aggregate complexity. It trains a large number of weak learners (constrained models) arranged in a sequence, such that each learner in the sequence learns from the mistakes of the learner before it. Then boosting combines all the weak learners in the sequence into a single strong learner.

Bagging works by training a large number of strong learners (relatively unconstrained models) arranged in a parallel pattern and then combining them to optimize their predictions.

- Regularization: Regularization refers to a large range of methods (L1/L2 regularization, Lasso regularization, dropout, etc.) but we won't go into details. The main idea of these methods is that they introduce a "complexity penalty" to the model. If the model wants to avoid incurring that penalty, it needs to focus on the most prominent patterns which have a better chance of generalizing well. Hence, it identifies and reduces the noise and complexity within the data.

b. Underfitting

Definition: Underfitting happens when a data model is unable to capture the relationship between the input and the output variables accurately, generating a high error rate on both the training set and unseen data.

Detection: High bias and low variance are good indicators of underfitting. This behavior can be observed while using the training dataset so underfitting is easier to identify than overfitting.

Prevention:

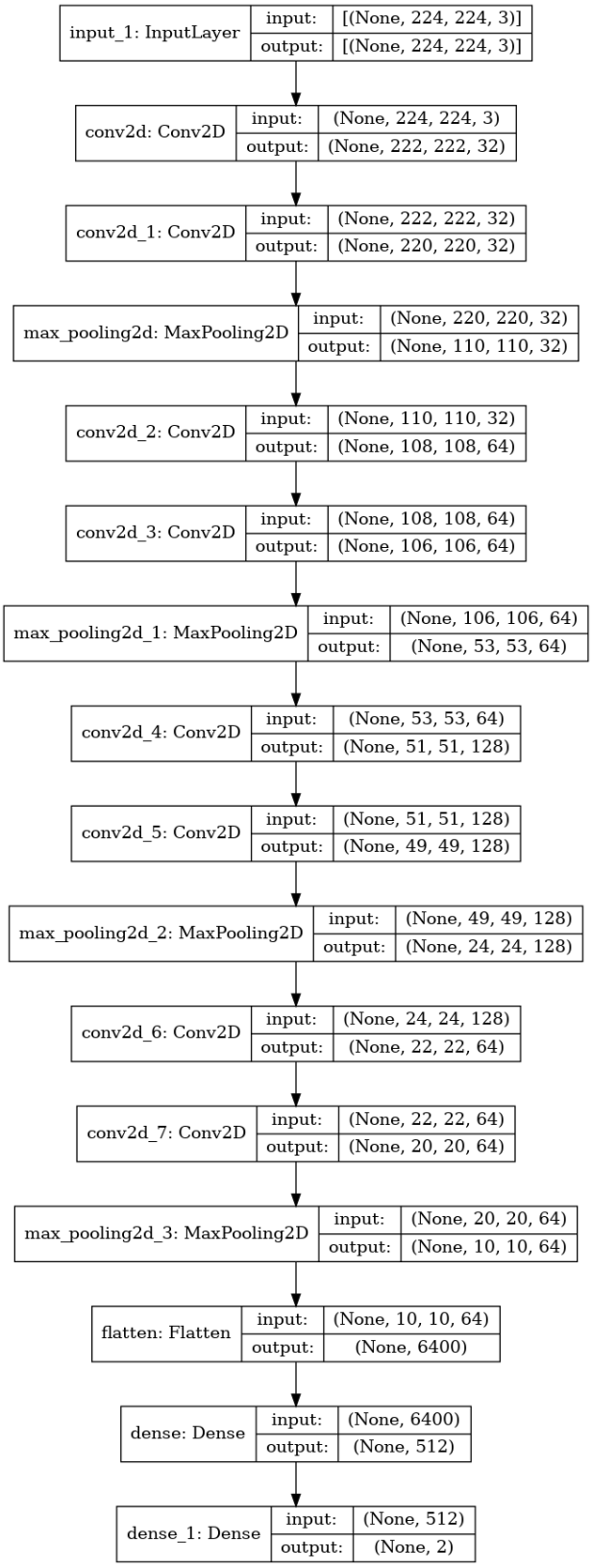
- Reduce regularization: If the data after regularization becomes too uniform, the model is unable to identify the dominant trend, leading to underfitting. By decreasing regularization, more complexity and variation is introduced into the model, allowing for better training of the model.
- Increase the duration of training: Stopping training too soon can result in underfitting so by extending the duration of training, it can be avoided.
- Feature selection: In contrast to overfitting, an underfitted model may be caused by the lack of features that can help the model detect the relevant patterns to make accurate predictions. Adding features and complexity to the dataset can help overcome underfitting and yield better training results.

3. Model description

a. Components

- 2D Convolutional layer:
  A convolutional layer's parameters consist of a set of learnable filters (2-dimensional arrays of weights). Every filter is spatially small but extends through the full depth of the input volume. A filter in the 2-dimensional convolutional layer convolves over the input data, performing an elementwise multiplication. As a result, it will summarize the results into a single output pixel. The filter will perform the same operation for every location it slides over, transforming a 2-dimensional matrix of features into a different 2-dimensional matrix of features, called a feature map. Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, or eventually more complex patterns on higher layers of the network.
- Pooling layer:
  A pooling layer is the layer added after the convolutional layer. The pooling layer operates upon each feature map separately to create a new set of the same number of pooled feature maps. Pooling involves applying a pooling operation to feature maps. In our model, we use maximum pooling (or max pooling) as the pooling operation; it will calculate the maximum value for each patch of the feature map. The size of the pooling operation is smaller than the size of the feature map. This means that the pooling layer will summarize the features detected in the input into a downsized set of pooled feature maps, which progressively reduces the spatial size of the representation and the amount of parameters and computation in the network, and hence, also control overfitting.
- Fully connected layer:
  Neurons in a fully connected layer have full connections to all activations in the previous layer. Their activations can then be computed with a matrix multiplication followed by a bias offset. The use of fully connected layer is to compile the data extracted by previous layers to form the final output.

b. Structure


input_1: InputLayer
input: [(None, 224, 224, 3)]
output: [(None, 224, 224, 3)]
conv2d: Conv2D
input: (None, 224, 224, 3)
output: (None, 222, 222, 32)
conv2d_1: Conv2D
input: (None, 222, 222, 32)
output: (None, 220, 220, 32)
max_pooling2d: MaxPooling2D
input: (None, 220, 220, 32)
output: (None, 110, 110, 32)
conv2d_2: Conv2D
input: (None, 110, 110, 32)
output: (None, 108, 108, 64)
conv2d_3: Conv2D
input: (None, 108, 108, 64)
output: (None, 106, 106, 64)
max_pooling2d_1: MaxPooling2D
input: (None, 106, 106, 64)
output: (None, 53, 53, 64)
conv2d_4: Conv2D
input: (None, 53, 53, 64)
output: (None, 51, 51, 128)
conv2d_5: Conv2D
input: (None, 51, 51, 128)
output: (None, 49, 49, 128)
max_pooling2d_2: MaxPooling2D
input: (None, 49, 49, 128)
output: (None, 24, 24, 128)
conv2d_6: Conv2D
input: (None, 24, 24, 128)
output: (None, 22, 22, 64)
conv2d_7: Conv2D
input: (None, 22, 22, 64)
output: (None, 20, 20, 64)
max_pooling2d_3: MaxPooling2D
input: (None, 20, 20, 64)
output: (None, 10, 10, 64)
flatten: Flatten
input: (None, 10, 10, 64)
output: (None, 6400)
dense: Dense
input: (None, 6400)
output: (None, 512)
dense_1: Dense
input: (None, 512)
output: (None, 2)

Layers which have the type ‘Conv2D’, ‘MaxPooling2D’, ‘Dense’ are convolutional layers, pooling layer, and fully connected layer respectively. Flattening is the process of converting the data into a 1-dimensional array in order to input it to the next layer (fully connected layer).

The input image has a width of 224px and a height of 224px. After that, we use 2 convolutional layers and a pooling layer (max pooling) to reduce the dimensions of the feature maps. The pooling layer summarizes the features present in a region of the feature map generated by a convolutional layer. We add 3 more sets of this combination (2 convolutional layers with a max pooling layer) to our model. Before moving to the last step, we must flatten the data first. Lastly, we add 2 fully connected layers to produce the final output and by that, the model is built completely.

c. Advantages and disadvantages

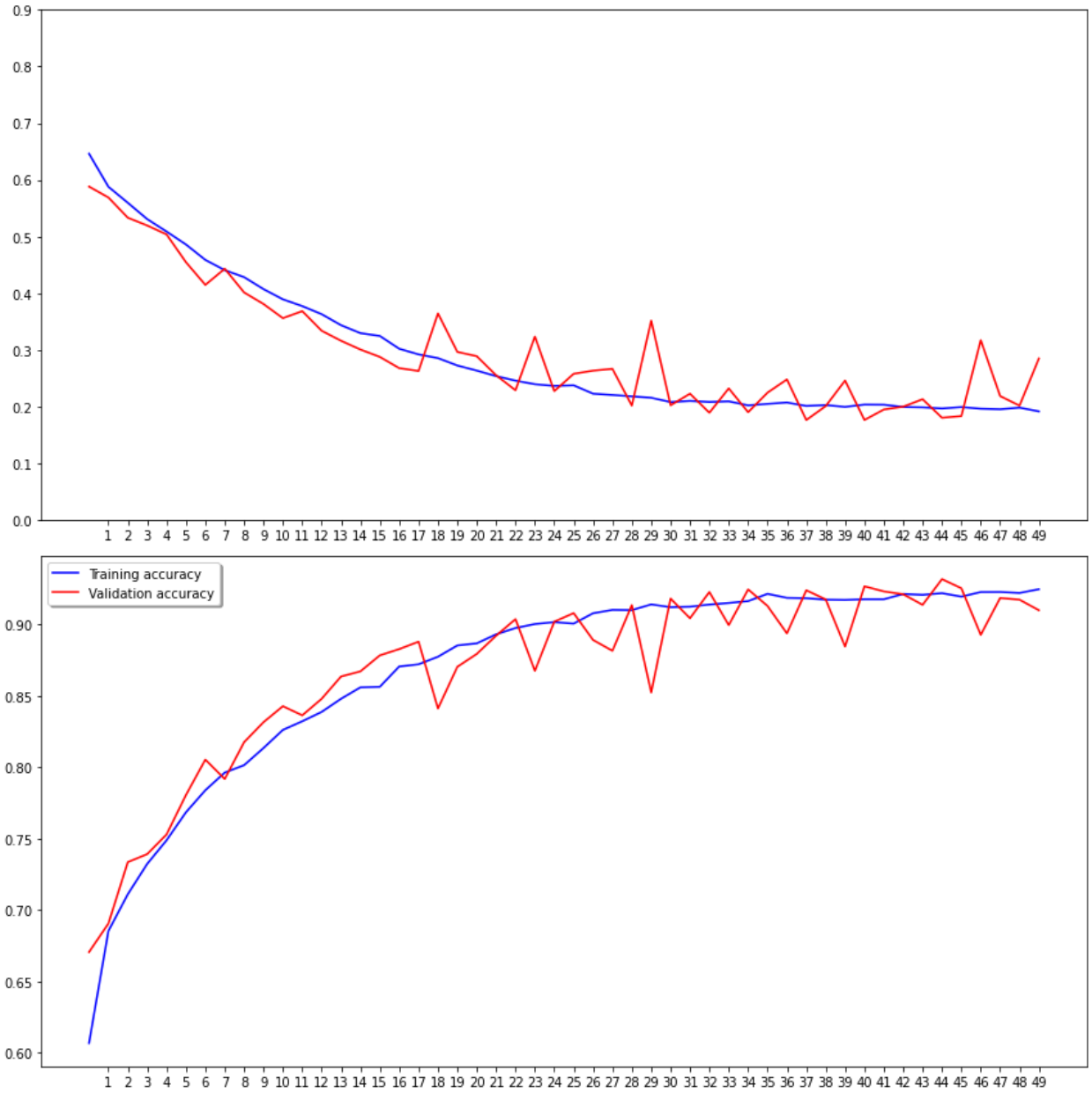
- Advantages:
  Our model is quite easy to implement, and it uses only 3 basic layers but still achieves relatively high accuracy.
- Disadvantages:
  It takes our model quite a lot of time to reach an accuracy of over 90%.

d. Why we choose this model

After implementing and training different kind of models, we conclude that this model is the most suitable one, because it has the highest accuracy among all versions, and it does not overfit the training data much. Also, it does not use dropout layer or batch normalization layer, which are not permitted in the scope of this project. We will mention more about them in the improvement section.

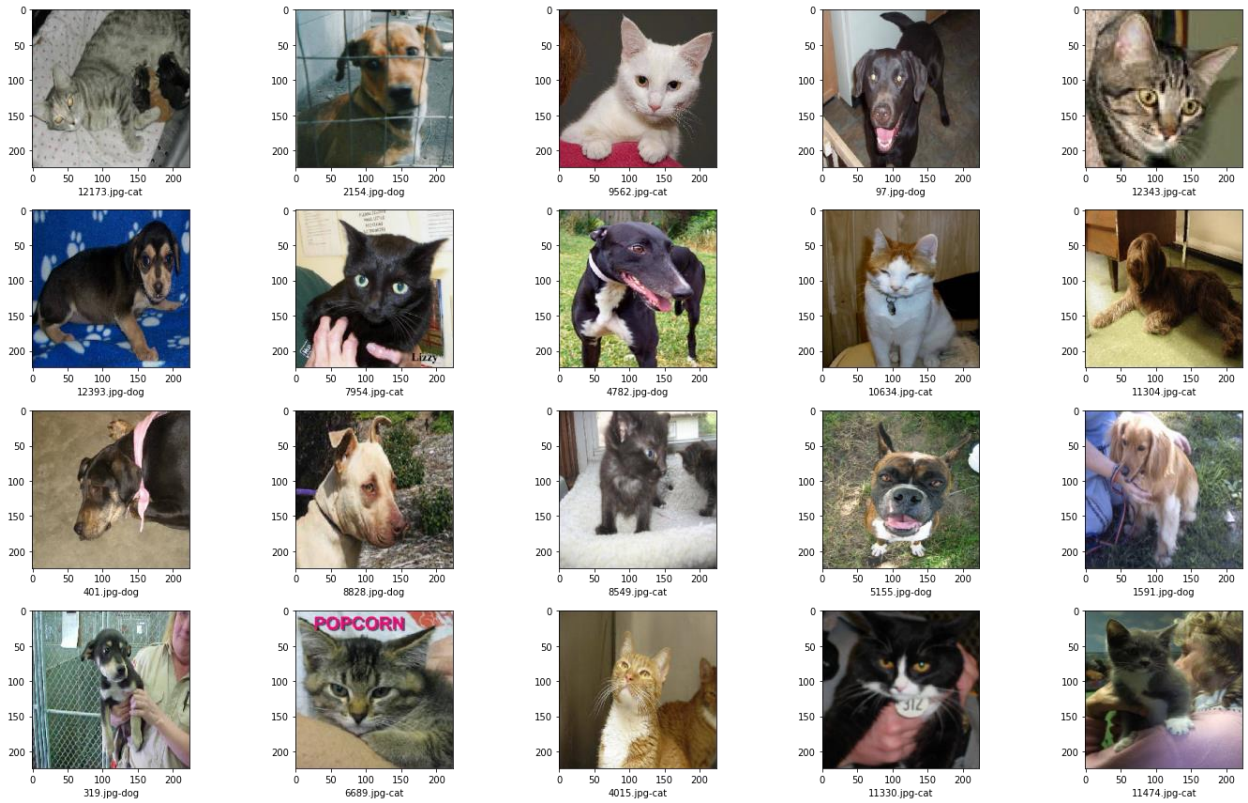
4. Results

a. Visualization



The first graph shows the loss overtime and the second one shows a comparison between training accuracy and validation accuracy. As you can see from the graph, our model has an accuracy of more than 90 percent, and it does not overfit the training data much.

Prediction samples



b. Explanation

For the optimizer, we use RMSprop, which have adaptive learning rate and utilize mini batch gradient descent. Mini-Batch Gradient Descent can improve convergence speed but introduce spikes to our graph because some mini-batches have 'by chance' unlucky data for the optimization. We started out with learning rate of 0.0001 and no momentum.

c. Improvement

We can make some improvements to our model by adding 2 more layers: dropout layer and batch normalization layer

- Dropout layer:
  Dropout works by probabilistically removing, or “dropping out” inputs to a layer, which may be input variables in the data sample or activations from the previous layer. It has the effect of simulating a large number of networks with very different network structure and, in turn, making nodes in the network generally more robust to the inputs. It is known as a regularization technique that prevents neural networks from overfitting.
- Batch normalization layer
  Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.

5. References

Training and test sets
Validation set
Overfitting
Underfitting
Convolutional Networks
Beginner-friendly Project Cat and Dog classification using CNN
Build a Convnet for Cat vs Dog Classification
How to Classify Photos of Dogs and Cats (with 97% accuracy)
How to Reduce Overfitting with Dropout Regularization in Keras
A Gentle Introduction to Batch Normalization for Deep Neural Networks
Keras CNN Dog or Cat Classification
Hey Siri!! is it a Cat or Dog?