

# Dogs Classification by Deep Learning

Guoxing Yao

March 2021

## 1 Abstract

This project trains a deep learning model to classify a set of dog images into five breeds. First, the whole dataset is fetched from Google search by a browser extension, then it is normalized and tried a overfitting model. In the next step, the whole dataset is split into training, validation and testing datasets, taking 70%, 21% and 9% respectively. After the model is trained with a series of different augmentation parameters, the optimal values include rotation = 90, zoom = 0.4, and shear = 0.6. Following these values, regularization is also employed to get the best values of L2 = 0.001, Dropout = 0.5 and no Batch normalization. Finally, a set of pre-trained model are utilized based on these parameters and NASNet produces the best accuracy eventually. Comparatively, model built in this project fail to beat all pre-trained models. The whole project evaluates the model performance by learning curve by plotting their loss and accuracy values.

## 2 Introduction

Deep Learning is becoming the dominant method to classify images since it is able to contribute a much higher accuracy compared with the conventional machine learning methods for a big dataset. Since Alexnet took revolutionary award by deep learning model in ImageNet competition 2012, more and more applications start to utilize deep learning. In the medical system, deep learning are used to diagnose the Pneumonia[1], and corona virus[2].

In this project, a deep learning model by Tensorflow/Kera on Google Colab is built to categorize a collective of dog images into four categories with high accuracy. All these images are stored in Google drive accessible from Google colab.

In this report, Section 3 described how dataset is pre-processed including normalization and display sample images, Section 4 described data splitting. From Section 5 on, several methods are employed to achieve the best performance. Section 5 tried a group of augmentation parameters, Section 6 tried regularization and Section 7 uses the pre-trained models including VGG16, NASNet, DenseNet and RESNet.

### 3 Data Preparation

#### 3.1 Data Description

The whole dataset contains 3176 images collected in batch from Google Search by a chrome extension, *Download All Images*, and most of them are colored. After the invalid, duplicate and grey images are removed manually, they are categorized into five groups, Canis lupus, Golden retriever, German shepherd, Husky and Poodle. Their distribution are displayed in Figure 1 and table 1.

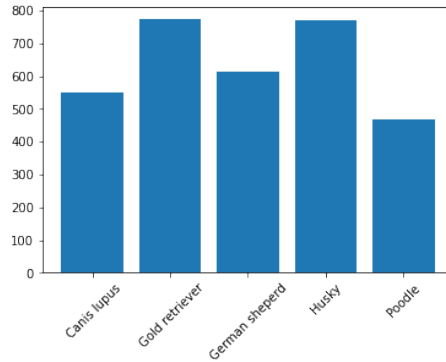


Figure 1: Dog images distribution by category

Dog categories	Amount of images
Canis lupus	550
Golden retriever	773
Germen shepherd	614
Husky	770
Poodle	469
<b>Total</b>	<b>3176</b>

Table 1: Distribution of dog images

Apparently, images in 5 categories are distributed relatively balanced.

#### 3.2 Data normalization and visualization

After all data are loaded, each image file is to be re-scaled to the range  $[0, 255]$  and dimension to  $(256, 256, 3)$ . They are naturally converted in 3 channel values, following the RGB data format. Some dog images are sampled in Figure 2.

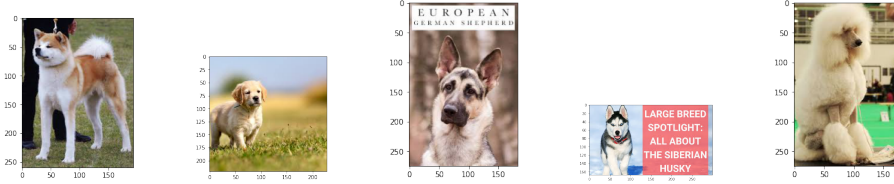


Figure 2: Dog sample images in each category

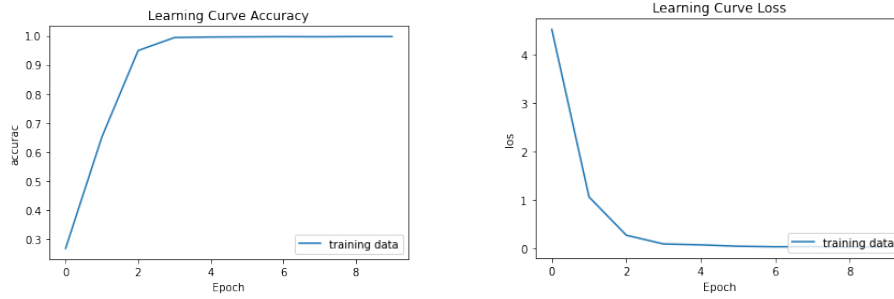


Figure 3: Dog sample images in each category

### 3.3 Overfitting model

In this subsection, all dataset are trained to get a overfitting model, shown in Figure 3.

## 4 Split dataset and model training

Different from overfitting section holding all images, the datasets is split into three subsets, training, validation and testing datasets from this section on. The whole datasets is split into training and validation sets by the ratio of 70% vs 30% first, then the testing set is split from validation set by 70% vs 30%. These files are selected by a set of random numbers stored locally. Table 2 shows the distribution of three sub datasets.

To test model performance between divergent model structures, including different filters, with early stop on validation dataset and accuracy with on testing dataset by evaluation, several experiments are designed in table 3 and table 4.

Both table 3 and 4 have maximal layers of 6 because more layers would lead to errors. In both table 3 and 4, accuracy values vibrate at the second experiment, but keeping an incremental trend as the layers increases. Two tables got similar accuracy values. In both tables, overfitting exists and as shown in table 4, the last one achieves the best solution with layers of [16,16,16,16,16,16]

dataset	Total	Category	Amount
training set	2225 (70%)	Canis lupus	385
		Golden retrieve	542
		German shepherd	430
		Husky	539
		Poodle	329
validation set	663 (21%)	Canis lupus	115
		Golden retrieve	161
		German shepherd	128
		Husky	161
		Poodle	98
testing set	288 (9%)	Canis lupus	50
		Golden retrieve	70
		German shepherd	56
		Husky	70
		Poodle	42

Table 2: Split dataset into training, validation and testing

## 5 More parameters by data generator

Data augmentation is a powerful tool for deep learning by a sequence of image processing[5]. In this project, data augmentation is also utilized to achieve a better result. To compare the trend with the best architecture configuration with three parameters, (I) data rotation range is valued of 45, 90, 135 and 180, (II) zoom from 0.2, 0.4, 0.6 and 0.8, (III) shear range from 0.2, 0.4, 0.6, 0.8, following the optimal design from Section 4. It is assumed that these parameters has no correlations between. Their learning curve is summarized in the table 5, 6 and 7.

The best result is at the 2nd design with rotation = 90, while it failed to obtain the trend between linear increase rotation range values.

The best result got is at the 3th experiment with zoom of 0.4, while it failed to obtain the trend between linear increase rotation range values.

The best result got is at the 3rd experiment with shear range of 0.6, while it failed to obtain the trend between linear increase rotation range values.

## 6 Regulation Effect

Besides the different neuron network architecture parameters above, different regulation parameters are also utilized to tune the model performance. There are three design below, (I) batch normalization in table 7, (II) drop out = [0.25, 0.5, 0.75] in table 8, and (III) L2 regularization = [0.1, 0.01, 0.001], in consecutive order.

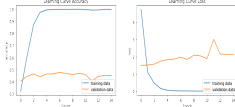
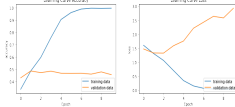
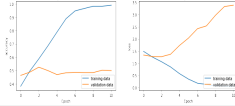
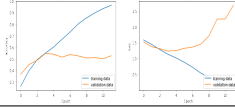
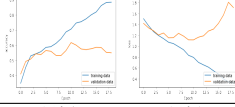
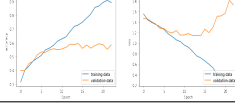
	Learning Curve	Neurons in each layer	loss accuracy
1		(32,5)	2.0346 0.5069
2		(32,32,5)	2.8531 0.4757
3		(32,32,32,5)	3.1409 0.5104
4		(32,32,32,32,5)	2.2817 0.5486
5		(32,32,32,32,32,5)	1.6675 0.5729
6		(32,32,32,32,32,32,5)	1.6175 0.6076

Table 3: Experiments to tune layers I

In table 8, the Batchnormalization is used in each convolutions layer, excluding the dense layer, but failed to achieve better performance in this experiment as well as worsen the overfitting issue. From now on, batch normalization is cast off in later steps.

Following the experiment in table 8, the dropout values are applied. But it failed to display a linear trend and produce overfitting results to some degree. The best accuracy is obtained at dropout = 0.5 and picked for the next experiment.

Following the table 9 , a sequence of L2 parameters 0.1, 0.01, 0.001, 0.005 are used. The model with L2 = 0.001 get the best solution.

## 7 Use pretrained models

One important application of Deep Learning is to apply existing models to the new dataset, three models are utilized, VGG16, ResNet50, DenseNet and NasNet, by removing their dense layer and modify input parameters. Their performance are listed in table 11.

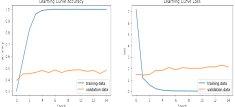
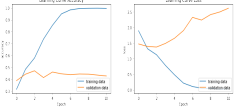
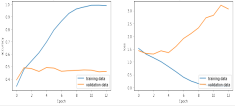
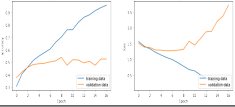
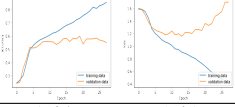
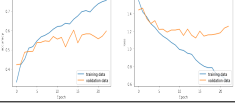
	Learning Curve	Neurons in each layer	loss accuracy
1		(16,5)	2.0357 0.5208
2		(16,16,5)	2.2803 0.4896
3		(16,16,16,5)	3.0186 0.5
4		(16,16,16, 16,5)	2.7471 0.5486
5		(16,16,16, 16,16,5)	1.5836 0.5694
6		(16,16,16, 16,16,16,5)	1.1179 0.6285

Table 4: Experiments to tune layers II

In the table 11, the experiment continues from the optimal solution with dropout = 0.5, rotation = 90 and zoom = 0.4 and no batch normalization, the best accuracy is from the NASNet model and the worst one is from ResNet Large model. In addition, NASNet pre-trained model also beats all experiments designed before. The RESNet model achieved the worst result and even bad than model designed before.

## 8 Conclusion

This project is initialized to categorize 3176 dog images into five categories by the deep learning model. After the images are normalized, split into training, validation and testing subsets, model training is launched as the objective of the project. The dominant procedure is simple but time-consuming by tuning different parameters. After these parameters are processed in consecutive steps, the optimal layers = [16, 16, 16, 16, 16, 16], zoom = 0.5, shear = 0.6, rotation = 90, no batch normalization, L2 = 0.001, dropout = 0.5 and the NASNet model picked as the optimal solution is achieved.

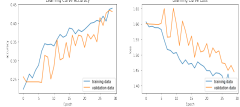
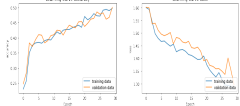
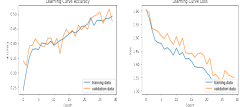
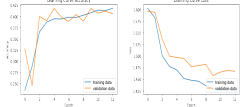
	Rotation	Learning Curve	Loss Accuracy
1	45		1.3944 0.4688
2	90		1.2825 0.5208
3	135		1.3279 0.4896
4	180		1.4476 0.4201

Table 5: Experiments to tune rotation range values

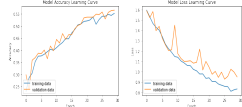
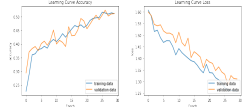
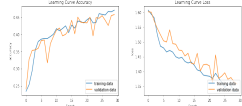
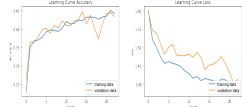
	Zoom	Learning Curve	Loss Accuracy
1	0.2		1.3944 0.4688
2	0.4		1.2826 0.5243
3	0.6		1.3385 0.45833
4	0.8		1.3902 0.4306

Table 6: Experiments to tune zoom values

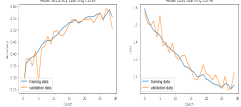
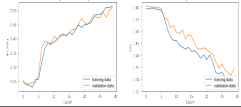
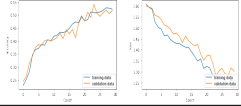
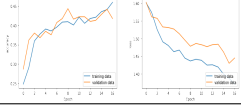
	Shear Range	Learning Curve	Loss Accuracy
1	0.2		1.3944 0.4688
2	0.4		1.3000 0.5417
3	0.6		1.2547 0.5625
4	0.8		1.4274 0.4236

Table 7: Experiments to tune Shear Range values

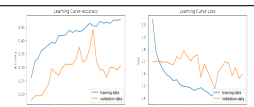
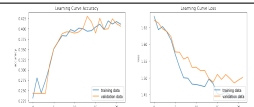
	BatchNormalized	Learning Curve	Loss Accuracy
1	Yes		0.5699 0.3125
2	No		1.4650 0.4167

Table 8: Experiments to tune Batch Normalization



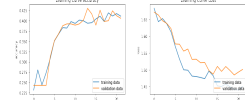
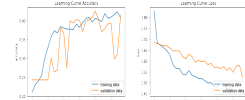
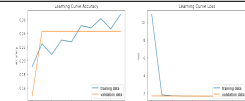
	dropout value	Learning Curve	Loss Accuracy
1	0.25		1.4650 0.4167
2	0.5		1.5109 0.4306
2	0.75		1.6714 0.2431

Table 9: Experiments to tune Dropout values

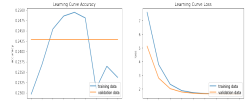
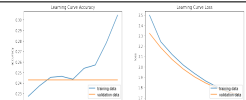

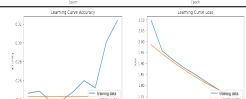
	L2 value	Learning Curve	Loss Accuracy
1	0.1		1.5990 0.2431
2	0.01		1.7283 0.2431
3	0.001		1.5109 0.4306
4	0.005		1.7372 0.2431

Table 10: Experiments to tune L2 values

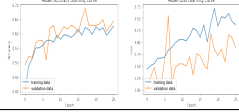
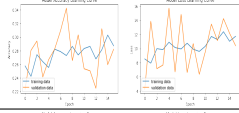
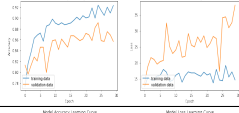
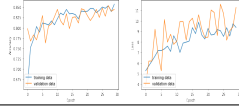
	Model Name	Learning Curve	Loss Accuracy
1	VGG16		3.1867 0.6111
2	ResNet50		9.9479 0.3194
3	NASNet Large		19.1832 0.9097
3	DenseNet121		8.6900 0.8646

Table 11: Experiments of different pre-trained models

## 9 References

1. E. Ayan and H. M. Ünver, "Diagnosis of Pneumonia from Chest X-Ray Images Using Deep Learning," 2019 Scientific Meeting on Electrical-Electronics Biomedical Engineering and Computer Science (EBBT), Istanbul, Turkey, 2019, pp. 1-5, doi: 10.1109/EBBT.2019.8741582.
2. El Asnaoui, K., Cgoohawki, Y. (2020). Using X-ray images and deep learning for automated detection of coronavirus disease. Journal of biomolecular structure dynamics, 1–12. Advance online publication.
3. Xie, S., Yang, T., Wang, X., Lin, Y. (2015). Hyper-class augmented and regularized deep learning for fine-grained image classification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2645-2654).
4. Baltruschat, I. M., Nickisch, H., Grass, M., Knopp, T., Saalbach, A. (2019). Comparison of deep learning approaches for multi-label chest X-ray classification. Scientific reports, 9(1), 1-10.
5. Zhang, Q., Liu, Y. (2018). Improving brain computer interface performance by data augmentation with conditional deep convolutional generative adversarial networks. arXiv preprint arXiv:1806.07108.