# Application of Deep Learning in Classification and Image Captioning

Tran Tien Dung, Pin Bang, Reagan Phung and Cuong Nguyen

Missouri State University

# Abstract

There are many computer vision problems in which deep learning has been implemented, including image classification, object detection, image colorization, image reconstruction and so on. Machine learning systems have been constructed to solve these problems and achieved notable success in many real-world applications. However, not many systems or applications aim to solve multiple problems at once or in conjunction with each other. Therefore, this project seeks to combine multiple application together as we believe only after these challenges and their solutions be combined, can more advanced artificial intelligent and human-like system be constructed. This project starts under the premise that image captioning, which is also a computer vision problem, will see massive bloom if implemented with virtual assistance, especially for children or visually impair people. Then, we combine image captioning with image classification to provide deeper information to the data provided by the technique. To set up the real-life application, we implement our proposed method in the Flickr30K and Stanford Dogs dataset.

# Table of Contents

# List of Figures

# List of Tables

## A. Introduction

In recent years, Deep learning (DL) techniques have become the main parts of various state-of-the-art multimedia systems and computer vision [1]. Roughly speaking, DL (or also known as representative learning) is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. The concept of DL was first mentioned by Warren S.McCulloch and Walter Pitts in 1943 [2] and then quickly become one of the hottest research topics in computer programming. Until now, after a long time of development, DL has proved itself to be efficient in various kind of application such as natural language processing [3] [4], speech and audio recognition [5] [6] and image classification [7]. To this regard, there has been a wide range of practical works done in this field, especially for classification and recognition purposes. However, almost all of the current works concentrate on improving a deep learning model to deal with a single task in a running time and hence it might not meet the expectation of an actual problem in real life. Therefore, in this project we aim to make an application by taking advantage of multiple deep learning techniques, which has capacity to perform more than one task at the same time. More particularly, the main contribution of this paper is to create a simple convolution neural network (CNN) which can classify the dog breeds from a fine-grained image and then employing recurrent neural network (RNN) to generate a brief description for the given data. We then run the created model in a real-life mobile application.

The motivation for this project is to prove the contribution of deep learning for human society. For example, in term of dog breeds this kind of application can help to find lost dogs, correcting the mislabel of images as well as breed determination of mixed breed dogs for health and safety

[8]. in this project, our experiment is done in the dog breed dataset, but it obviously can be further extent into some other problems which might be crucial for people, such as automatic system to find lost children in a big city.

The rest of this paper is designated as following: section B gives a summary of related theoretical background in this project; section C provides information about related work; section D reviews out implementation process; section E demonstrate the deployment and result of the system; section F is about conclusion and remark.

# B. Theoretical Background

## I. Neural Network Introduction

In the twenty-first century, the study of intelligence demonstrated by machine has experienced intense growth of interest. This success in the field of artificial intelligent (AI) owns many achievements to its subset machine learning, which in turn sees substantial development due to advancement in computer power, data collection and related theoretical knowledge. In other words, previously invented computational logic and system which were limited by the technology of their time have now been researched, implemented, and gained significant accomplishment.  One of such computational logic is artificial neural network (ANN), or neural network (NN) for short.

The architecture of neural network was inspired and partly mimics the biological construction of human brain [9]. The network consists of computational units called neuron, or node.

Each node, after receiving input information from either other nodes or an external source, will perform data calculation and produce an output. Associated with every input is a weight (w), which signifies its relative importance in comparison to other inputs. Multiple nodes are divided into different collective group called layers. After receiving the weighted sum of input, the node will apply various function on the sum to produce a result.

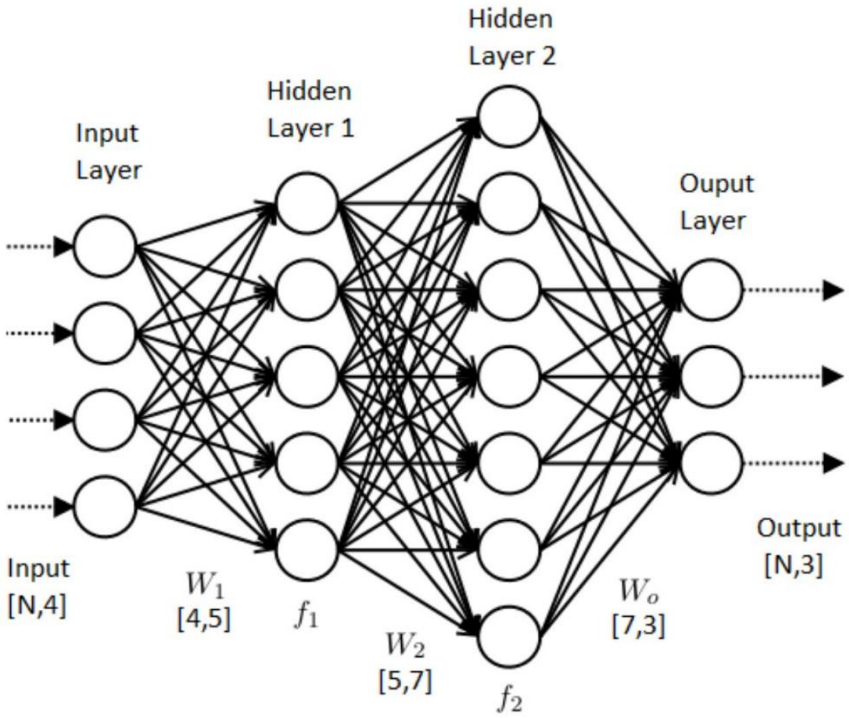


*Figure 1: Example of an ordinary neural network*

A traditional neural network usually includes the following components:

- Input Nodes (input layer): The layer that receives information from external source and pass them to other layers.
- Hidden nodes (hidden layer): The layers where intermediate computation is processed.
- Output Nodes (output layer): The layer that conforms to a desired output format for the neural network.
- Connections and weights: Each node transfer information to other node through a connection, which is assigned with a particular weight.
- Activation function: the activation function to be applied by a node on the weighted sum of input to produce a result.

The abundance of complicated task requires various network architecture to be implemented, from simple ones, like Perceptron or Feed Forward, to complex one like Deep Belief Network or Generative Adversarial Network.

In this project, we will be taking advantage of the following network architectures: Convolutional Neural Network (CNN) and Recurrent Neural Networks (RNN).

## II. Convolutional Neural Network

The traditional neural network does not perform well in image processing. If each pixel were to be used as an input value for such network, even a small image of size 224 x 244 and 3 channels would generate more than 150 thousand parameters to be trained. This may lead to numerous problems, including severe computational cost or overfitting. Therefore, working with image requires an advanced neural network type, called Convolutional Neural Network.

CNN can be constructed in many ways, thus vastly differs from the traditional network. However, the most notably difference is the introduction of several layers specialized in processing the image.

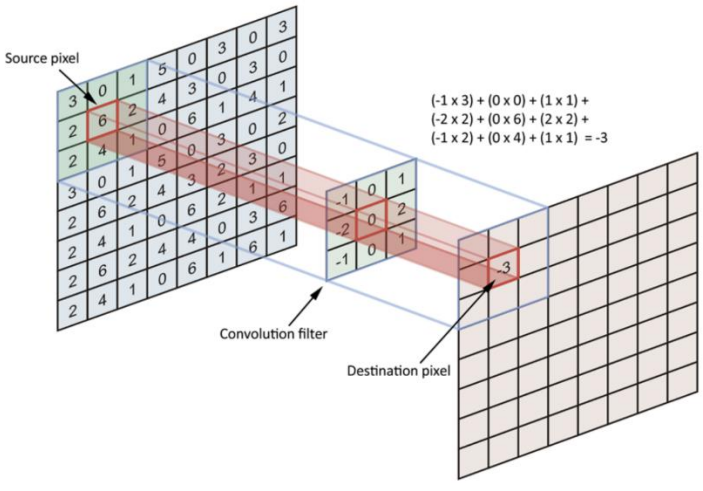


*Figure 2: The convolution operation*

One of such layers is the convolutional layer. This layer applies a filter to the original image by moving a kernel from top left to bottom right and employing convolution operation. This task brings about beneficial effects as:

- The number of parameters to learn by the network is reduced, as a kernel functions by exercising many-to-one mapping.
- Proximity is considered, meaning related or shared information between nearby pixels will be taken into calculation.

Another special layer is the pooling layer, which function like a convolution layer, but perform specific function like max pooling or average pooling, which takes the maximum or average value in the filter region, respectively.

## III. Recurrent Neural Network

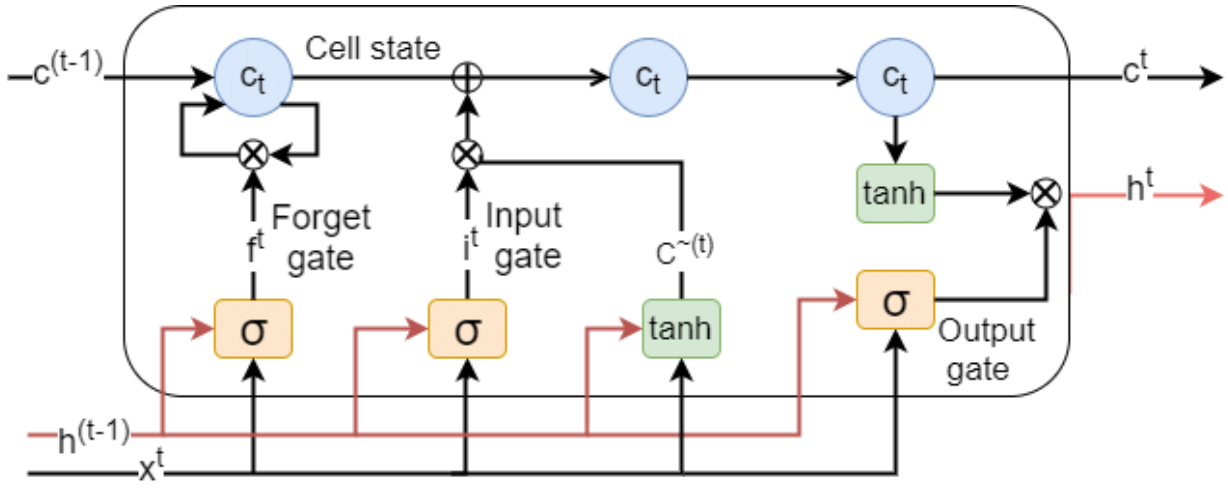


*Figure 3: LSTM unit Illustration*

Like the case of CNN, Recurrent Neural Network was constructed to solve certain problems that the traditional neural network falls short, specifically ones involve processing sequence data, like time series prediction, speed recognition, natural language processing…

RNN introduces the concept of memory. In RNN, the output of some layers will be used as input for previous layers, thus allowing information to persist.

Among the popular RNN architecture there is Long short-term memory (LSTM) architecture. A common formation of LSTM unit is composed of a cell and three regulators. The cell acts as the memory, saving different arbitrary values through time. The three regulators, which include an input gate, an output gate and a forget gate, control the flow of information in and out of the cell.

## C. Related Work

There has been a great deal of recent works in using deep learning for image classification and image captioning. Our work is specially designed to handle the fine-grained categorization, which is able to handle the dataset with a large number of classes.

Firstly, to build the model for classifying image, we take advantage of the work [9] in which the Authors propose a simple convolutional neural network and implement it in the MNIST data set [10]. After comparing the result with other methods in image classification, this paper proved that CNN gives a relatively good performance while it is much easier to be implemented. Besides, the work [11] also shared the same idea with our project is to make a mobile application to classify dog breeds; however, their proposed method is heavily based on the special characteristics of each dog breed. As a result, their approach is easier to be confused when it comes to hybrid dogs such as Husky + Corgi or Husky + Golden.

Secondly, to generate the auto caption for the dog image, we refer to the work of Andrej Karapathy [12] who proposed a model that generates natural language descriptions of images and their regions. Furthermore, in the work [13], the Authors combine deep convolutional networks for image classification with recurrent networks for sequence modeling and finally create a single network that can generates descriptions of images.

In the final result of our project, we mix the output of image classification and auto captioning to get the comprehensive descriptions for the dog images.

# D. Implementation

## I. Dataset

In this project, we have experienced with three different datasets, for image caption training and image classification.

- For image caption tasks, we use the Flickr30k dataset, containing 31,783 images and 158,915 English captions (5 for each image). These images and captions display people participating in daily activities and events [14].
- For image classification task, we used two datasets. One is the Stanford Dogs Dataset which contains 20,580 images from 120 species of dogs [15] [16]. The other is the 225 Bird Species dataset contains images of 225 species of birds, including 31316 images for training images, 1125 images for testing and 1125 images for validation [18].

## II. Image Captioning

### 1. Data preprocessing

Both the images and captions will need to be preprocessed before training.

Image will be used as an input to our network model; therefore, it must be passed in the form of a vector. For this task, we used the pretrain NASNet-Large CNN model in Keras. The model has been trained on more than a million images from the ImageNet database and can classify images into 1000 object categories [19].

We proceed to remove the last layer of the pretrained model so that we can use the model to extract a 4032-length vector from every image in the dataset.

Caption is something we want to predict, but it will be done so word by word. Therefore, we need to encode each word into a vector. We also will be removing some words that are uncommon from the caption dataset. All in all, after preprocessing, we ended up with 5437 different words in the corpus, which are indexed from 1 to 5437.

Then we create two dictionaries for the words, one to handle converting a word to its index, and the other to convert an index to its corresponding word.

## 2. Data Preparation

*Figure 4: Sample Train and Test Images*

Supposedly we have two training images, one depicts a “dog running on grass”, while the other shows a “cat running on water”. We want to generate a model that can correctly caption the image of a “dog running on water”.

First, we convert the two training images into their corresponding 4032-length vector, which are named “Image_1” and “Image_2” respectively.

Then we build the vocabulary for the training caption after adding two tokens “startseq” and “endseq” to the sentence:

Caption 1: startseq dog running on grass endseq

Caption 2: startseq cat running on water endseq

Vocab = {dog, cat, endseq, grass, on, running, startseq, water}

Then we assign an index for each word: dog – 1, cat – 2, endseq – 3, grass – 4, on – 5, running – 6, startseq – 7, water – 8.

What we want to achieve is after providing image vector and the first word as an input, the model can output the second word. Then by providing the image vector and the first plus second word, the model can output the third word. And so on.

| | X - Input | | Y – Output |
|---|---|---|---|
| Index | Image feature-vector | Partial Caption | Expected Word |
| 1 | Image_1 | [startseq] | dog |
| 2 | Image_1 | [startseq; dog] | running |
| 3 | Image_1 | [startseq; dog; running] | on |
| 4 | Image_1 | [startseq; dog; running; on] | grass |
| 5 | Image_1 | [startseq; dog; running; on; grass] | endseq |

*Table 1: Training data corresponding to one image and caption*

However, because the training process involves numeric data, we will pass the index for each word instead. And, as required by Tensorflow/Keras, the input data needs to be of the same length, so we pad zero at the end of each sequence, up to a predefined number for maximum caption length (which is 74 in this project).

| | X - Input | | Y – Output |
|---|---|---|---|
| Index | Image feature-vector | Partial Caption | Expected Word |
| 1 | Image_1 | [7; 0; 0; 0; 0; 0; 0; 0] | 1 |
| 2 | Image_1 | [7; 1; 0; 0; 0; 0; 0; 0] | 6 |
| 3 | Image_1 | [7; 1; 6; 0; 0; 0; 0; 0] | 5 |
| 4 | Image_1 | [7; 1; 6; 5; 0; 0; 0; 0] | 4 |
| 5 | Image_1 | [7; 1; 6; 5; 4; 0; 0; 0] | 3 |

*Table 2: Sequence as indexes with zeroes padding*

Finally, each word/index will be mapped to a pre-trained GLOVE word embedding model. This representation ensures words with similar meaning will have a similar representation.

## 3. Image Captioning Model

Finally, we use Keras Functional API to construct a model with the following architecture:

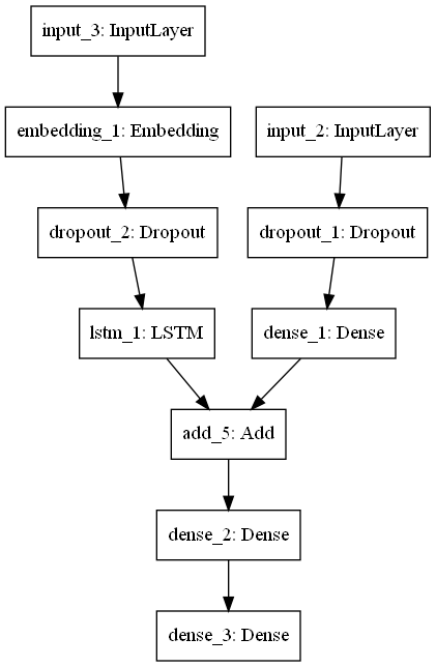


*Figure 5: Model architecture for Image Captioning*

The model summary is as follow:

| **Layer (Type)** | **Output Shape** | **Param #** | **Connected to** |
|---|---|---|---|
| input_3 (InputLayer) | (None, 74) | 0 | |
| input_2 (InputLayer) | (None, 4032) | 0 | |
| embedding_1 (Embedding) | (None, 74, 200) | 1087600 | input_3[0][0] |
| dropout_1 (Dropout) | (None, 4032) | 0 | input_2[0][0] |
| dropout_2 (Dropout) | (None, 74, 200) | 0 | embedding_1[0][0] |
| dense_1 (Dense) | (None, 512) | 2064896 | dropout_1[0][0] |
| lstm_1 (LSTM) | (None, 512) | 1460224 | dropout_2[0][0] |
| add_5 (Add) | (None, 512) | 0 | dense_1[0][0]<br>lstm_1[0][0] |
| dense_2 (Dense) | (None, 512) | 262656 | add_5[0][0] |
| dense_3 (Dense) | (None, 512) | 2789694 | dense_2[0][0] |
| Total params: 7,665,070<br>Trainable params: 6,577,470<br>Non-trainable params: 1,087,600 | | | |

*Table 3: Image Captioning Model Summary*

Then we proceed to train the model using specified dataset.

## II. Image Classification

### 1. Failed Attempt

At first, we started with training a model for bird images classification. We used the model Inception ResNet V2 from Keras for training from the beginning (no pretrained weight). After 20 epochs, we achieve an accuracy of around 80%.

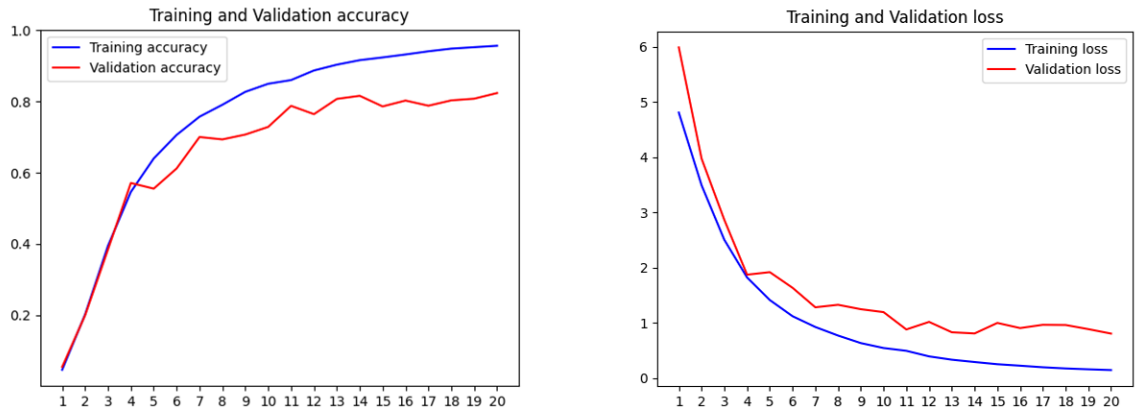


*Figure 6: Training and Validation for Bird Classification Model*

Therefore, our model for bird classification may potentially work well along with captioning task. However, we had to abandon the model due to its inability to work with the captioning model.

Then, we begin training the model for dog images classification, using the same model Inception V3 from Keras, and train every parameter, without pretrained weight. However, around the $10^{th}$ epochs, the model starts over fitting, with an accuracy of around 40%.

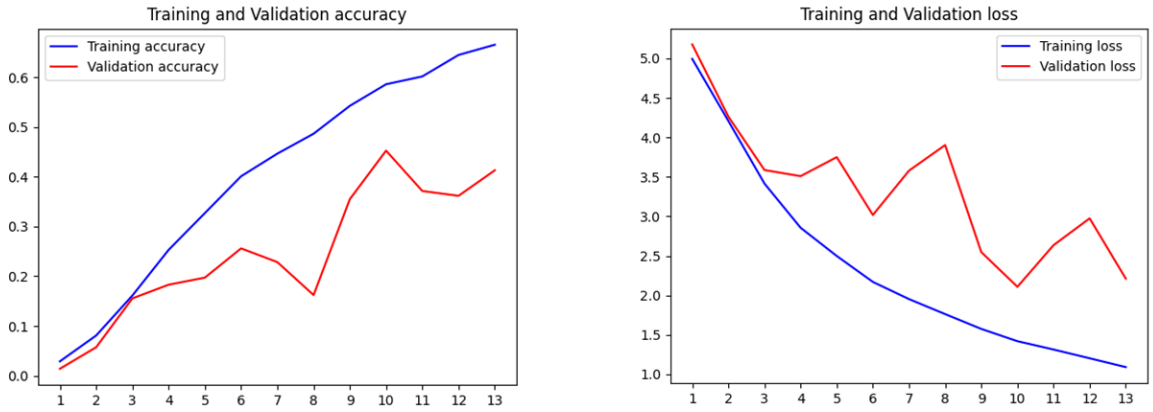


*Figure 7: Training and Validation for Dog Classification model without pretrained weight*

At this point, we decided to switch to a model with pretrained weight but add extra layers of our own.

## 2. Dog Image Classification Model

We also used the pretrained weight of Keras NASNet Large model for the dog image classification task. However, we add several extra layers right after the output of NASNet model.

With the layer “activation_260” as the last layer of NASNet large, our added extra layers architecture is as follow:

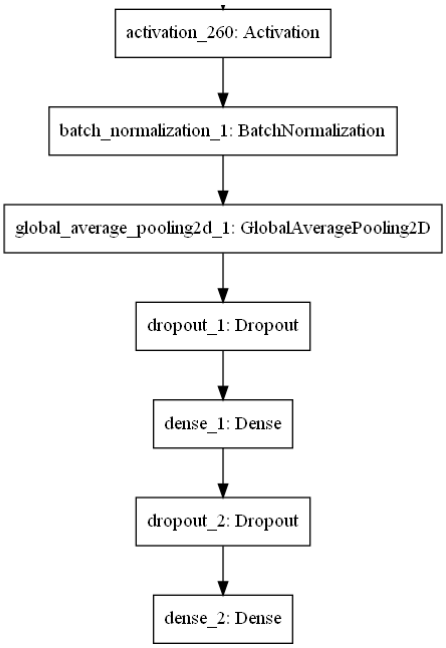


*Figure 8: Extra layers added to the NASNet Large model*

And the summary information for extra added layers is as follow:

| **Layer (Type)** | **Output Shape** | **Param #** | **Connected to** |
|---|---|---|---|
| activation_260 (Activation) | activation_260 (Activation) | 0 | normal_concat_18[0][0] |
| batch_normalization_1 | (None, 11, 11, 4032) | 16128 | activation_260[0][0] |
| global_average_pooling2d_1 | (None, 4032) | 0 | batch_normalization_1[0][0] |
| dropout_1 (Dropout) | (None, 4032) | 0 | global_average_pooling2d_1[0][0] |
| dense_1 (Dense) | (None, 1024) | 4129792 | dropout_1[0][0] |
| dropout_2 (Dropout) | (None, 1024) | 0 | dense_1[0][0] |
| dense_2 (Dense) | (None, 120) | 123000 | dropout_2[0][0] |
| Total params: 89,185,738<br>Trainable params: 4,260,856<br>Non-trainable params: 84,924,882 | | | |

*Table 4: Dog Image Classification model - Extra layers summary*

Then we proceed to train the model

Only after 4 epochs did the model each and accuracy of around 92%.

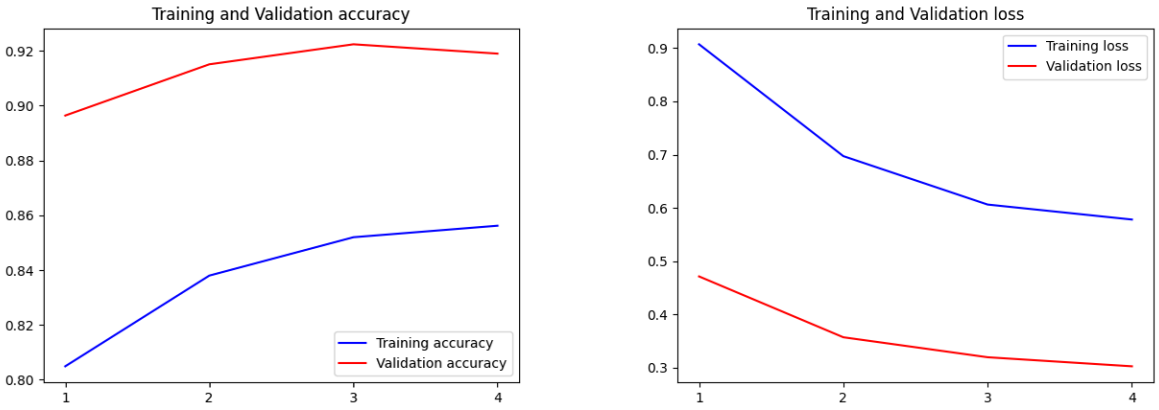


*Figure 9: Training and Validation for Dog Classification model with pretrained weight*

At this point, seeing the accuracy rate is substantially high, especially compared to the model without pretrained weight, we decided to stop training and implement this model for dog image classification.

## E. Deployment and Result

### I. Deployment

We planned to deploy this system on Google Cloud Platform, then link to through Facebook so that the system can work with Facebook Messenger. When a message containing an image is sent to Facebook Messenger, it will be forwarded to our system, which in turn captions it, predicts dog breed if the image contains a dog, and returns the result to the user.

So, we need to set up three main services.

First, we deployed an intermediate application on Google App Engine. This application will act as a bridge, connecting Facebook Messenger Webhook to our prediction service. It receives the message from user. If the message contains an image, the application gets the image attachment, else if the message contains a link to an image, the application gets the image from that link.

Then the application proceeds to convert that image to a base64 string and forward it to the caption service. After received the response from the caption server, which should be a string, the intermedia application checks to see if such caption contains the word “dog”. If it does, the application assumes there is a dog in the provided image and send that image to the dog classification service. If the classification service can identify the dog breed, the application will specify the dog breed, instead of just “dog” in general, and return the result to Facebook Messenger.

The intermediate is programming using Golang.

Second, we deployed both the caption service and classification service on Google AI Platform, which is a Linux virtual machine running as a part of Google Compute Engine. These services communicate with the intermediate application through HTTP request.

## II. Result

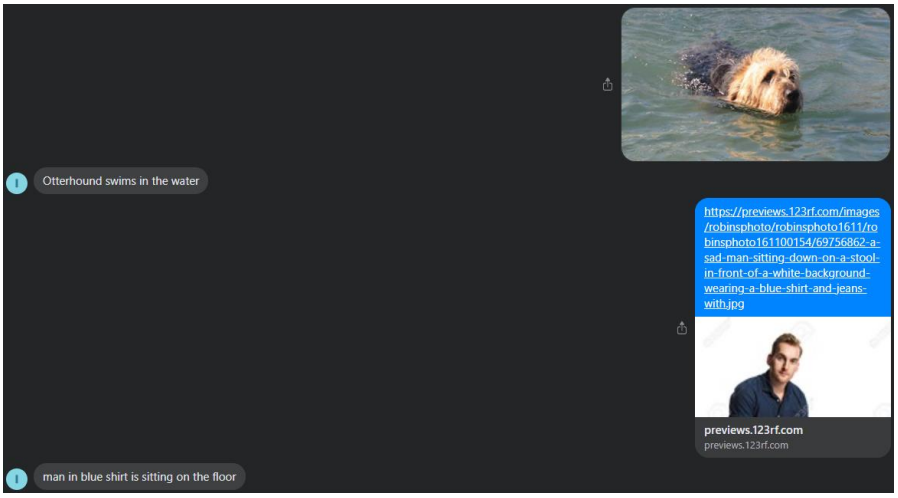


*Figure 10: Demonstration of the system result*

Our services have been deployed and it is currently online.

For dog image classification task, the system can correctly recognize the dog breed most of the time. However, for the image captioning task, the system may occasionally generate a partly incorrect caption. For example, the input image may depict a man sitting on a chair, but the system would predict he is sitting on the floor instead.

Despite its flaws, the system could function adequately to a certain degree, and has achieved our initial goals, that is combining multiple Machine Learning services together.

# F. Conclusion and Remark

## I. Conclusion

In this project, we attempted to combine multiple Machine Learning services together, namely image captioning and image classification.

The image captioning task involves using Recurrent Neural Network and Keras Functional API to build an image captioning model.

The image classification task combines pretrained Convolutional Neural Network model with extra layers to achieve a model with substantial accuracy.

After building the models, we deploy various services to make them function as real-world application.

There are rooms for improvement in three area: more accuracy caption prediction, more animal or object to be identified and better cloud service integration.

Overall, while the system may not work flawlessly and has some detriments, it is still serviceable and accurate up to a certain degree.

## II. Remark

During the deployment of this system, our team have attained considerable amount of knowledge.

Cuong Nguyen: I learned about how Recurrent Neural Network operates and Keras Functional API. Also, I have gained experience in building services to load the model, use it for prediction and implementing it on Google Cloud Platform services.

Dung Tran: I learned that loading pretrained weight and adding custom layers can massively increase accuracy compared to training the network from scratch. Also, the project allows me to work with Keras Functional API, which is something I never done.

Reagan Phung: Working with both CNN and RNN has cemented my knowledge for neural network. I gain a lot of experience in performing hyper parameter tuning, model generator, early stopping and many other things to assist the training process.

Bin Pang: This project gives me experience in working with complex neural network and grant me knowledge about available model in Keras. Also, I see how Machine Learning can be implemented to create real-world application.

# References


[1] H.-Y. Ha, Y. Yang, S. Pouyanfar, H. Tian and S.-C. Chen, "Correlation-based deep learn- ing for multimedia semantic concept detection," in *In International Conference on Web Information Systems Engineering*, 2015.

[2] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics,* vol. 5, no. 4, pp. 115-133, 1943.

[3] Y. Kim, "Convolutional neural networks for sentence classification. CoRR abs/1408.5882," 2014. [Online]. Available: http://arxiv.org/abs/1408.5882.

[4] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Conference on Empirical Methods in Natural Language Processing.*, 2013.

[5] K. Han, D. Yu and I. Tashev, "Speech emotion recognition using deep neural network and extreme learning machine," in *Interspeech. ISCA, 223–227*, 2014.

[6] X. Zhang and D. Wang, "Deep learning based binaural speech separation in reverberant environments," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing 25 (2017), 1075–1084.*, 2017.

[7] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Neural Information Processing Systems Conference*, 2012.

[8] X. Wang, V. Ly, S. Sorensen and C. Kambhamettu, "Dog breed classification via landmarks," in *IEEE International Conference on Image Processing (ICIP), Paris, 2014, pp. 5237-5241, doi: 10.1109/ICIP.2014.7026060.*, 2014.

[9] Y.-Y. Chen, Y.-H. Lin, C.-C. Kung, M.-H. Chung and I.-H. Yen, "Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in Demand-Side Management for Smart Homes," 2019.

[10] T. Guo, J. Dong, H. Li and Y. Gao, "Simple convolutional neural network on image classification," in *IEEE 2nd International Conference on Big Data Analysis (ICBDA), Beijing, 2017, pp. 721-724, doi: 10.1109/ICBDA.2017.8078730*, 2017.

[11] Y. Qiao, "THE MNIST DATABASE of handwritten digits," Retrieved 18 August 2013, 2007.

[12] J. Liu, A. Kanazawa, D. Jacobs and P. Belhumeur, "Dog Breed Classification Using Part Localization," in *7572. 172-185. 10.1007/978-3-642-33718-5_13.*, 2012.

[13] A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions," in *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 4, pp. 664-676, 1 April 2017, doi: 10.1109/TPAMI.2016.2598339*.

[14] O. Vinyals, A. Toshev, S. Bengio and D. Erhan, "Show and tell: A neural image caption generator," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 3156-3164, doi: 10.1109/CVPR.2015.7298935.*, 2015.

[15] P. Young, A. Lai, M. Hodosh and J. Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," *Transactions of the Association for Computational Linguistics,* pp. 67-68, 2014.

[16] A. Khosla, N. Jayadevaprakash, B. Yao and L. Fei-Fei, "Novel dataset for Fine-Grained Image Categorization," in *First Workshop on Fine-Grained Visual Categorization (FGVC), IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[17] J. Deng, W. Dong, R. Socher, K. L. L.-J. Li and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2009.

[18] G. Piosenka, "225 Bird Species," 31 07 2020. [Online]. Available: https://www.kaggle.com/gpiosenka/100-bird-species/.

[19] mathworks, "mathworks," [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/nasnetlarge.html. [Accessed Nov 2020].