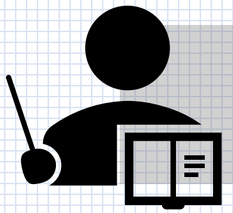
APPLIED PHYSICS 186 – ACTIVITY 18

MACHINE LEARNING:

Convolutional Neural Networks



Professor: Dr. Maricor Soriano
Instructor: Jayson Cubero

Rene L. Principe Jr.
2015-04622

dataset

Figure 1. Dataset for Kaggle containing images of cats and dogs [2].

For deep learning using images, CNN architectures are efficient in extracting features. The method uses the concept of convolution with designed filters. Before anything else, convolutional layers were visualized from scratch to understand how CNN extract features as shown in Figures 2 and 3.

In this report, we use packages from Tensorflow and Keras to implement the algorithm. The dataset chosen contains 12470 dog and 12476 cat images. Images were converted to grayscale, normalized, and resized to 100 x 100 dimensions. The dataset was divided such that 80% is set for training, while 20% is for testing. 20% of the training data was set for a simultaneuous validation. The network architecture used is summarized in Figure 4.

Visualizing CNN Layers

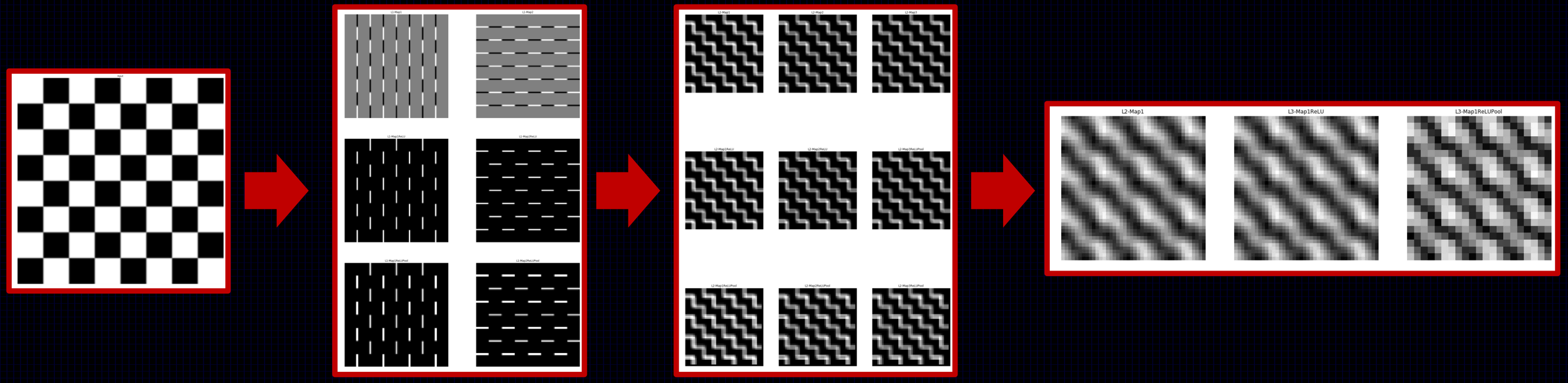


Figure 2. Each layer in the plot above shows one "convolutional + pooling layer". More layers show how deeper features are Extracted. The latest layer shall then be flattened to form a fully connected Neural Network.

Visualizing CNN Layers

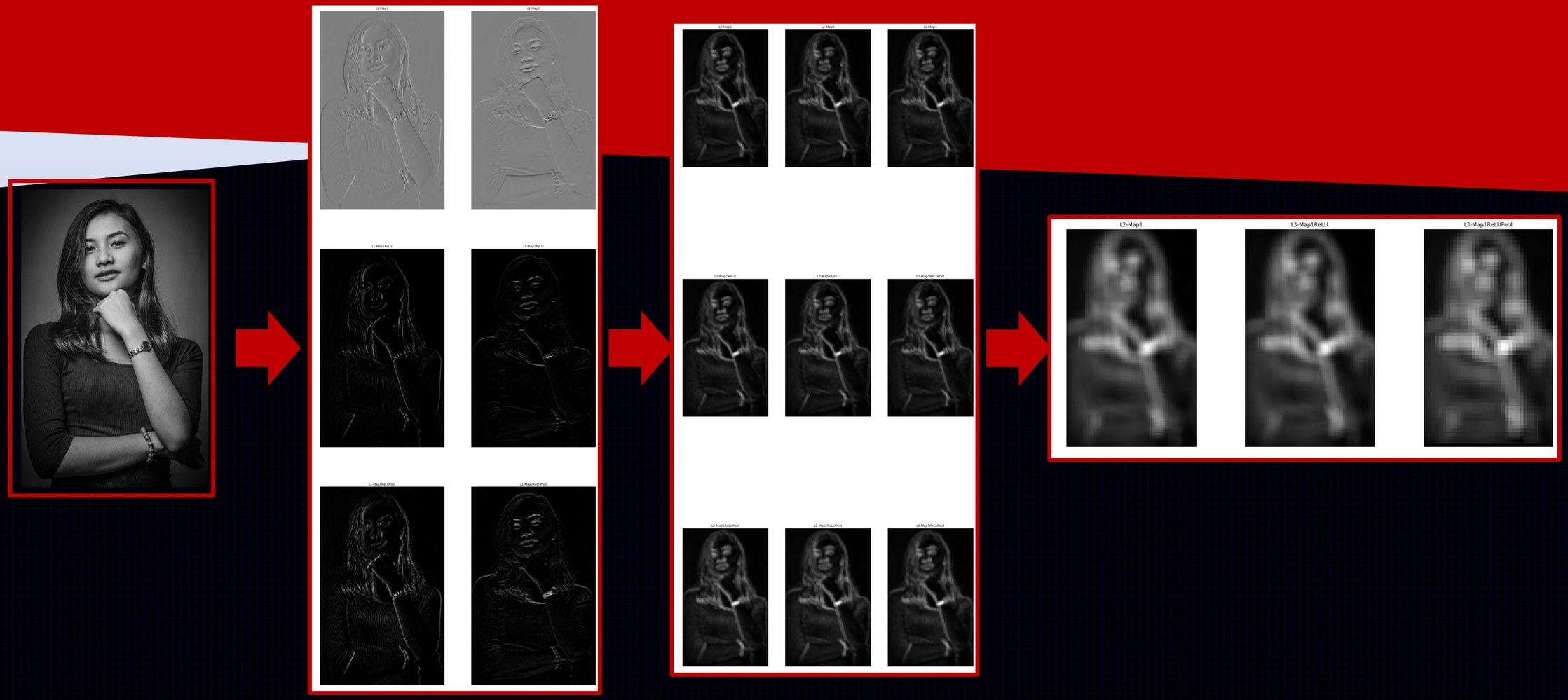


Figure 3. One of the most common applications for CNN is facial recognition as it extracts contours and textures as shown in the plots above. Here I used a portrait photograph of my friend which I shot in a studio.

```
Model: "sequential_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_26 (Conv2D)           (None, 98, 98, 64)        640
_________________________________________________________________
activation_52 (Activation)   (None, 98, 98, 64)        0
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 49, 49, 64)        0
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 47, 47, 64)        36928
_________________________________________________________________
activation_53 (Activation)   (None, 47, 47, 64)        0
_________________________________________________________________
max_pooling2d_27 (MaxPooling (None, 23, 23, 64)        0
_________________________________________________________________
flatten_13 (Flatten)         (None, 33856)             0
_________________________________________________________________
dense_26 (Dense)             (None, 64)                2166848
_________________________________________________________________
activation_54 (Activation)   (None, 64)                0
_________________________________________________________________
dense_27 (Dense)             (None, 1)                 65
_________________________________________________________________
activation_55 (Activation)   (None, 1)                 0
=================================================================
Total params: 2,204,481
Trainable params: 2,204,481
Non-trainable params: 0
```

Figure 4. Model architecture summary.

The network takes a grayscale image as input, has 2 “convolutional + pooling” layers and a dense fully connected layer which leads on to a single output value activated my a sigmoid function. After one iteration, the network will return a percentage value from 0 (dog) to 1 (cat). The training and validation loss and accuracies were tracked as shown in Figure 5. The model was trained for 30 epochs for a run time of ~1 hour and a final validation accuracy of 76% was obtained.

To verify, we predict the classes of the testing dataset using the weight parameters from training and the result was shown in Figure 6. Positive results were predicted on 76% of the 4989 labeled testing dataset. Sample prediction of 20 samples are shown in Figure 5.

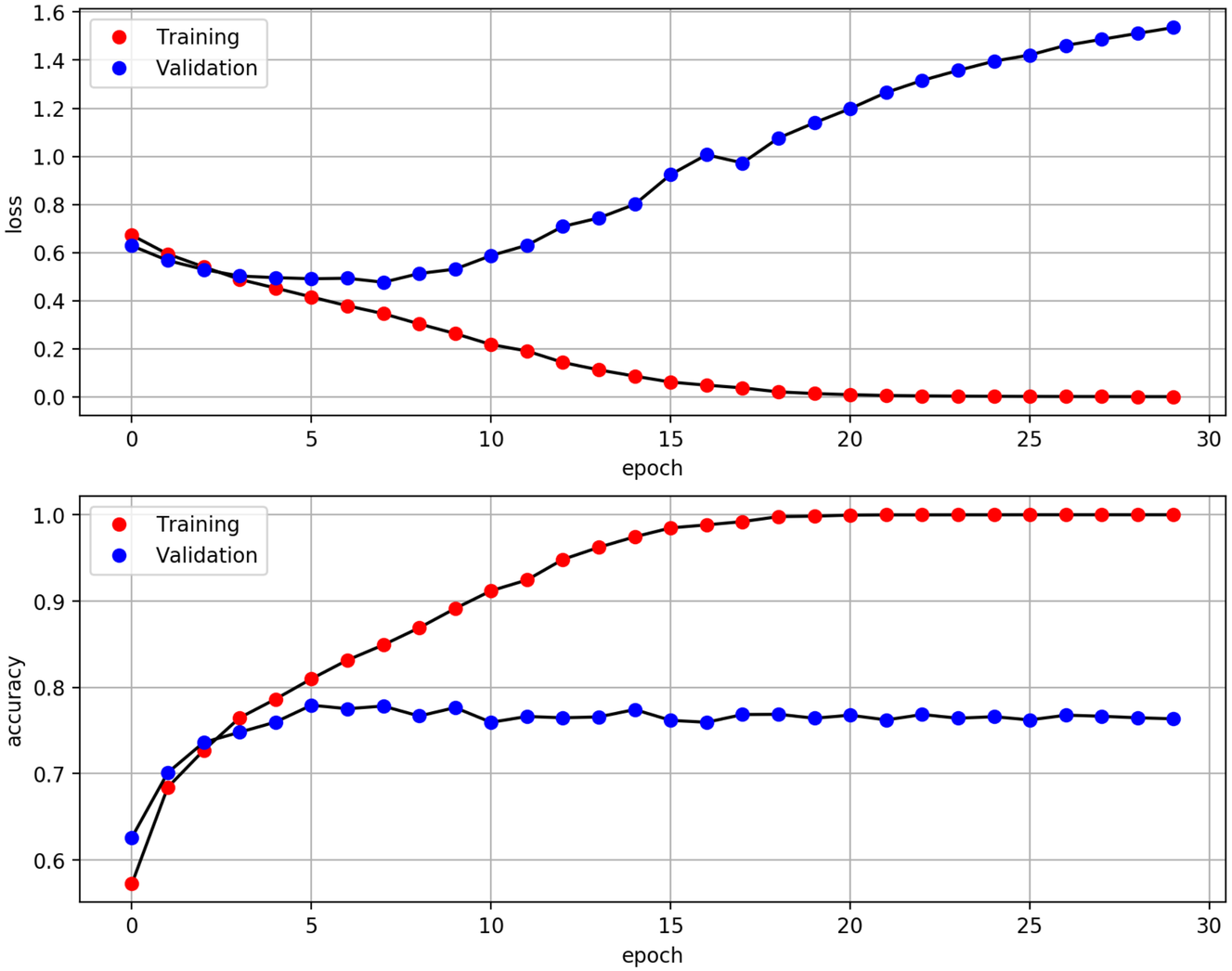


Figure 5. Model performance in terms of binary cross-entropy loss and prediction accuracy. Validation accuracy converged to a staple value of 0.76

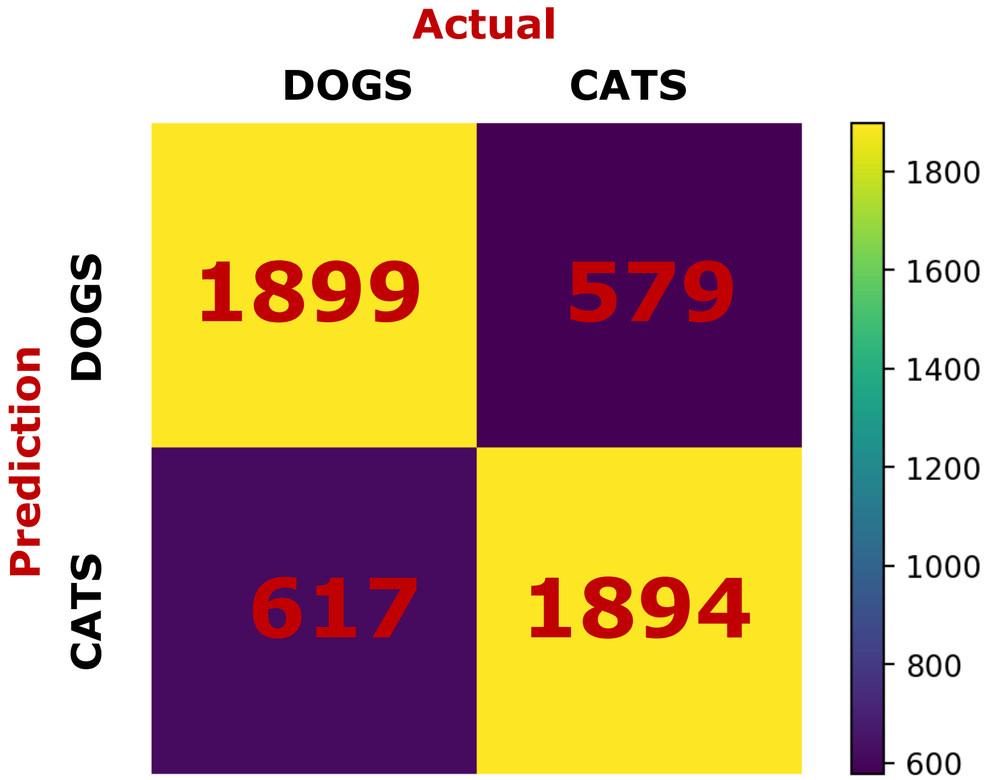


Figure 6. Confusion Matrix on the Testing dataset of 4989 samples.

Further model optimization can be done to yield better results. Another option is to try out deeper architectures and implement the analysis on three color channels.

I have successfully implemented CNN of grayscale images which yielded a 76% accuracy and for that, I'd give myself a rating of 11.

References:

[1] M. Soriano, "Convolutional Neural Networks", 2019.

[2] Kaggle DogVCat Competition: http://www.kaggle.com/c/dogs-vs-cats

Figure 8. Sample predictions of never-before-seen images show how the model can predict the class. Most of the predictions are correct, but there are still instances of failed prediction.