Capstone Project **Thao Phung**
Machine Learning Engineer Nanodegree **July 12, 2017**

I. Definition

Project Overview

Computer Vision has gained many attentions in recent years. Many applications have applied computer recognition for better user experience as well as human's life improvement. Convolution Neural Networks (CNNs) play a vital role for fast-improvement of computer vision. From a simple network that recognizes hand-written digits (Lenet) [1] to now, it can recognize different types of animals, human's faces for security [2], or it is a part of self-driving car. Here, in this project, I apply the CNNs to recognize, then count and classify sea lions out of pictures.

The Steller sea lions in the western Aleutian Islands are endangered with the population declined 94% in the last 30 years. NOAA Fisheries Alaska Fisheries Science Center pays a lot of attention and effort to conserve the kind by counting the population of sea lion in western Aleutian Islands every year. The center takes many photos of the regions using airplanes and unoccupied aircraft system, then manually count and classify sea lions. It takes up to four months every year. Computer Vision algorithms, especially CNNs, can help the NOAA biologists.

Problem Statement

The goal is to build a neural network to count and classify the sea lions from collected aerial images.

1. Download and preprocess the sea lion dataset
2. Train a convolutional neural network to recognize and classify sea lions in images
3. Test the convolutional neural network

The trained CNN can recognize and classify the sea lions.

Metrics

The error is calculated using Root Mean Squared Error (RMSE) which takes the average of the square of all of the error. RMSE is commonly used to measure the error for numerical prediction.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

n: number of categories
y_i: true vector (assuming one-hot encoding)
$\hat{y}_i$: predicted vector

II. Analysis

Data exploration

The dataset is given by NOAA Fisheries Alaska Fisheries Science Center. These are aerial photographs which are collected by airplanes and unoccupied aircraft systems. It consists of 948 images for training set. It the same 948 images with color dotted in another train dotted set. And there are 16,836 images in test set. The images in Train Dotted set are manually colored dotted over animals by NOAA biologists so it is human-labelled ground truth. The color scheme of the dots is:

- Red: adult males
- Magenta: subadult males
- Brown: adult females
- Blue: juveniles
- Green: pups

In both training and test sets, there are images that have black regions which are added by human to avoid double-counting. The network can ignore these black regions when making prediction. The images are colored and have around 3744 * 5616 pixels each.

Since the images are taken from aircraft, there are real natural Aleutian Islands' images which include sea, tree, sea lions and other kinds of animals.

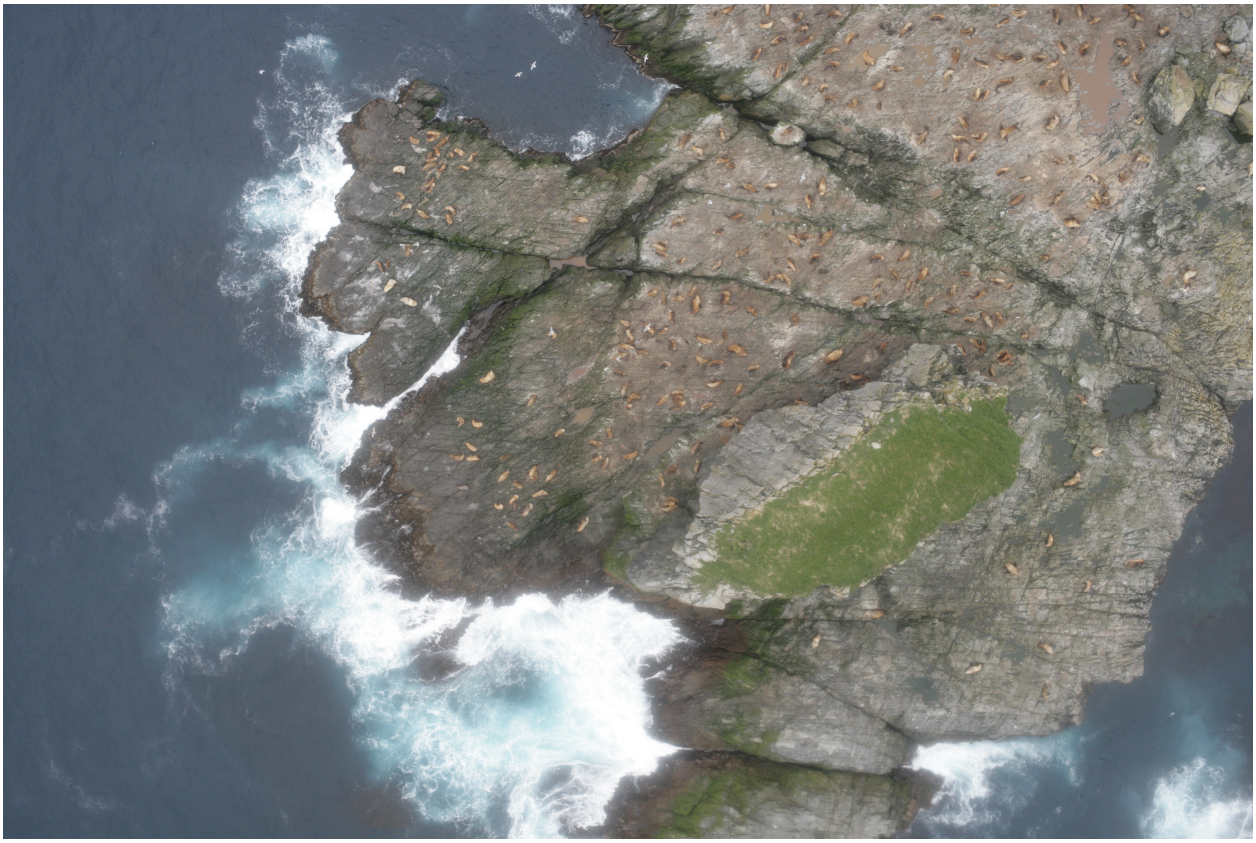

Figure 1: An aerial photograph in Train dataset

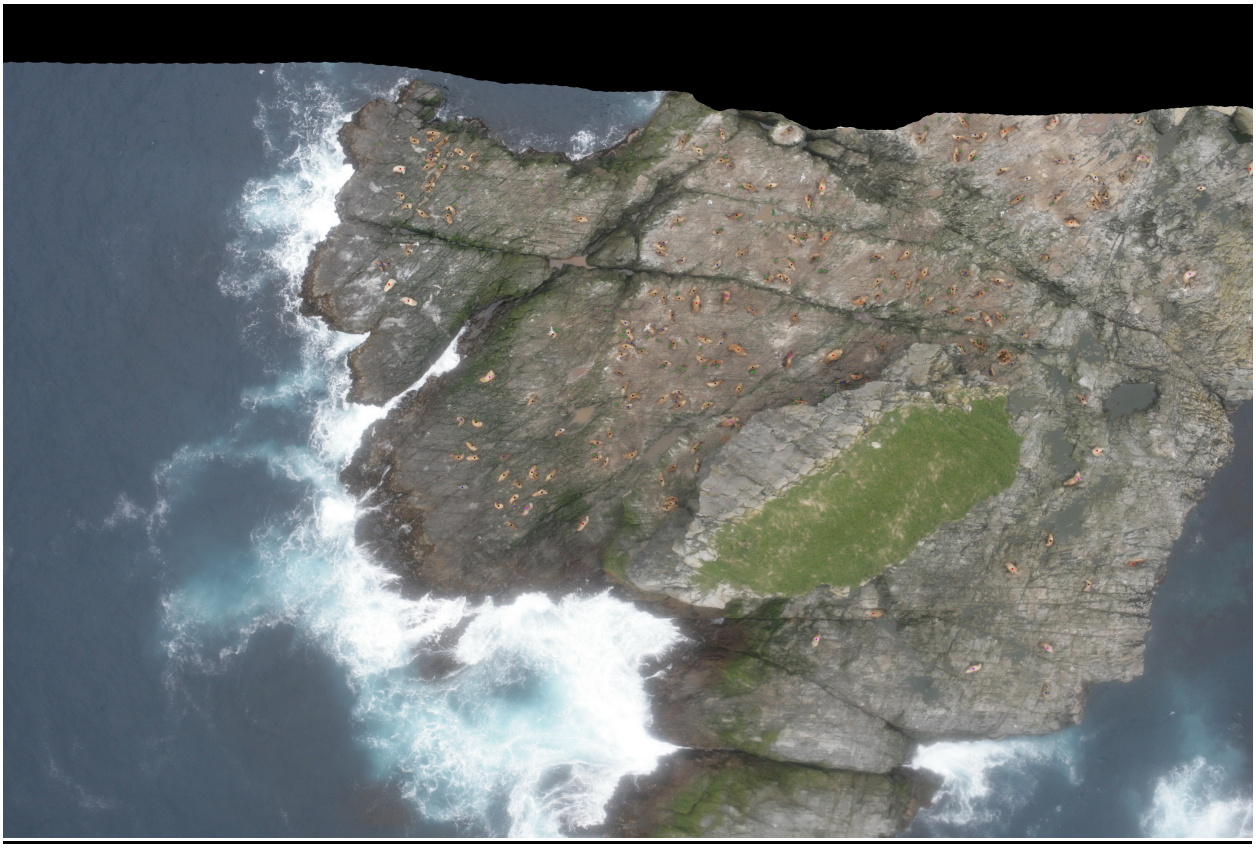

Figure 2: Image 1 with blackened region and colorful dots on sea lions

In training set and training dotted set, there are 58 images are mismatched so these images are removed out of both sets. After that there are 891 images in Train and Train Dotted sets.

91 images are taken out and put into validation set randomly.

I only use the training dataset to train and test the neural network because I can get the label after preprocessing the dotted images.

Algorithms and Techniques

1. Convolutional Neural network

Convolutional Neural Network (CNN) is very effective in image recognition and classification. It successfully identifies not only faces, objects but also traffic signs to help robots and self-driving cars work more properly with better vision ability. A CNN is a stack of layers (e.g. Convolutional layers, Pooling layers, Fully Connected layers) and made up of neurons which have learnable weights and biases, then perform dot product. The last layer of the networks is a single differentiable score function which produce the probabilities for each predicted value.

- Input

  An image can be represented as a matrix of pixel values. Besides width and height, the image has a parameter called channel which consists of the color values. If the image is colored, it will have 3 channel RGB (red, green, and blue) which of them has value in range 0 and 255 (0 is black, and 255 is white). If the

image is in grayscale, it will have only one channel. Therefore, the size of the matrix of an image will be [width * height * number of channel].

- Convolutional Layer

  The Convolution layer extracts features from the input image and preserves the spatial relationship between pixels. The input of convolutional layer is a matrix from the previous layer. If it is the first layer of the network, the input will be the raw image matrix of pixels. In convolutional layer, there is a filter which is a matrix. In Figure 3, the image (green) is 5x5 pixels, the filter (orange) is 3x3 pixels. The filter slides over the image and extracts the features. When sliding, the filter computes dot product and produces a new matrix called Convolved Feature or Feature Map. Each filter produces a different feature map that means each filter extracts different feature (e.g. edges, colors, curves). The more number of filters, the more image features get extracted and the better the network becomes at recognizing patterns in testing images. A filter which plays as feature detector has 3 parameters to control the size of output feature map: stride, depth, and zero-padding.

  - Depth: the number of filters used for the convolution operation
  - Stride: the number of pixels by which the filter matrix slide over the input at one time. When the stride is 1, then the filter jumps 1 pixel at a time. The larger the stride is, the smaller the feature map
  - Zero-Padding: allows us to control the size (width and height) of the output feature map.

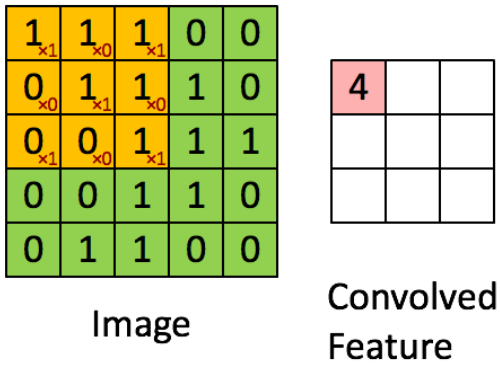

Figure 3: On the left, an 5x5 pixel image (green) and a 3x3 pixel filter (orange). The filter computes the dot products and produce a new matrix, Convolved Feature, on the right, value '4' is the first result of the first dot product computation of the first filter's slide over the image.

In general, the more convolutional layers, the more complicated featured are able to learn. However, it also produces more neurons and needs more memory.

- ReLU (Rectified Linear Unit)

  ReLU is a non-linear operation which applies element wise activation function (i.e. max(0, x) with thresholding at 0). It replaces all negative pixel values in the feature map (usually after convolutional and fully connected layers) by 0.

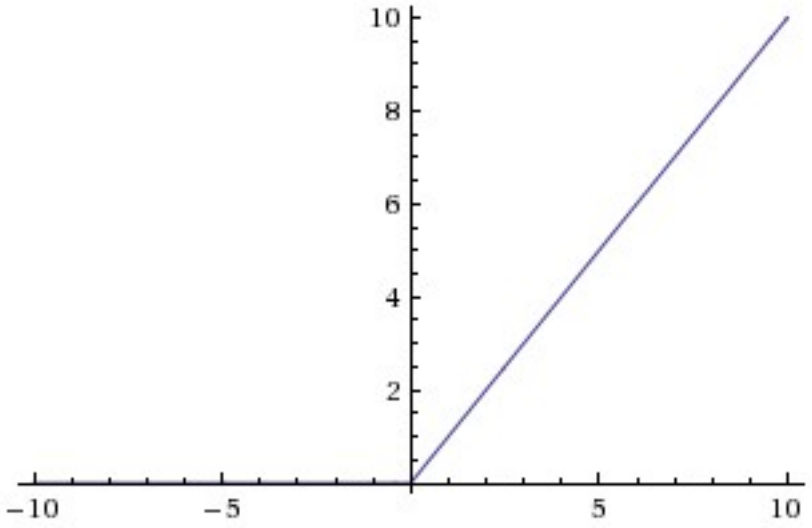

Figure 4: ReLU with max(0, x)

- MaxPooling Layer
  The purpose of pooling layer is to reduce the size of feature map but keeps the most important information. There are different kinds of pooling such as max, average, sum, etc. In this network, we use max pooling that retains the largest element from the rectified feature in each small region. The pooling layer has filter and stride parameters. In Figure 5, a filter of 2x2 pixels with stride of 2 slides over a feature map. In each 2x2 window, the operation keeps the largest pixel value. By this way, we can get smaller and manageable feature maps and also reduce the number of parameters and computations in the network.

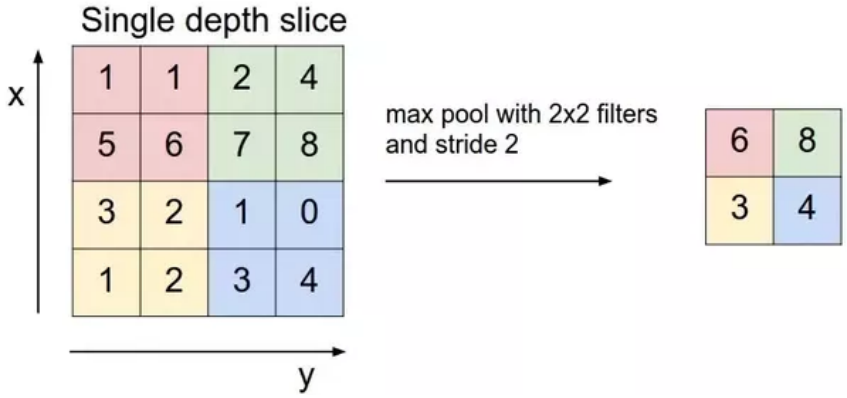

Figure 5: A filter of 2x2 with stride of 2 slides over a feature map. In each 2x2 window, the operation keeps the largest pixel value.

- Fully connected layer
  Fully connected layer connects every neuron in previous layer to every neuron in current layer. It takes in the features that are extracted by convolutional and pooling layers to classify the original images. It uses a softmax activation function to output the probabilities for classes. The final fully connected layer has the size

[1x1x number of classes]. In addition, the loss is also calculated by loss function. The loss is measured based on how well the output compared to the ground truth value. In another word, loss function measures the quality of set of parameters (weights and biases) that are used to produce the output.

2. **Optimizer**

Optimization helps to find the best weights to minimize the loss function by updating the weights during the training.

Benchmark

This was a competition on Kaggle and the winner got the RMSE of 10.85645 and the runner-up got 12.51. I would like to use these number as the range of my error.

III. Methodology

Data Preprocessing

1. Split the raw and dotted dataset into training set, validation set and test set randomly
2. Read both raw images and dotted image
3. Remove noise out of the image
4. Get the absolute difference between raw image and dotted image
5. Convert images from in color into in grayscale to detect the blob
6. Detect blobs
7. Based on the color of the dot, sea lions are added to their groups
8. Out 2 arrays, one consists arrays of images, one consists arrays of numbers of each types of sea lions.

Implementation

The pre-trained VGG16 is used. The architecture of VGG16 is in Figure 6. I keep all Convolutional blocks and add on top of that one Flatten layer and 2 Dense layers as Fully-connected classifier.

To implement the network,

1. Load the preprocessed dataset
2. Load the pre-trained VGG16 with 'ImageNet' weights
3. Add Flatten layer on top of the last convolutional block
4. Add the first Dense layer, apply LeakyReLU activation function
5. Add the last Dense layer which gives the output, apply Linear activation function
6. Fit the model with training and validation datasets.
7. Save weights and history of the training

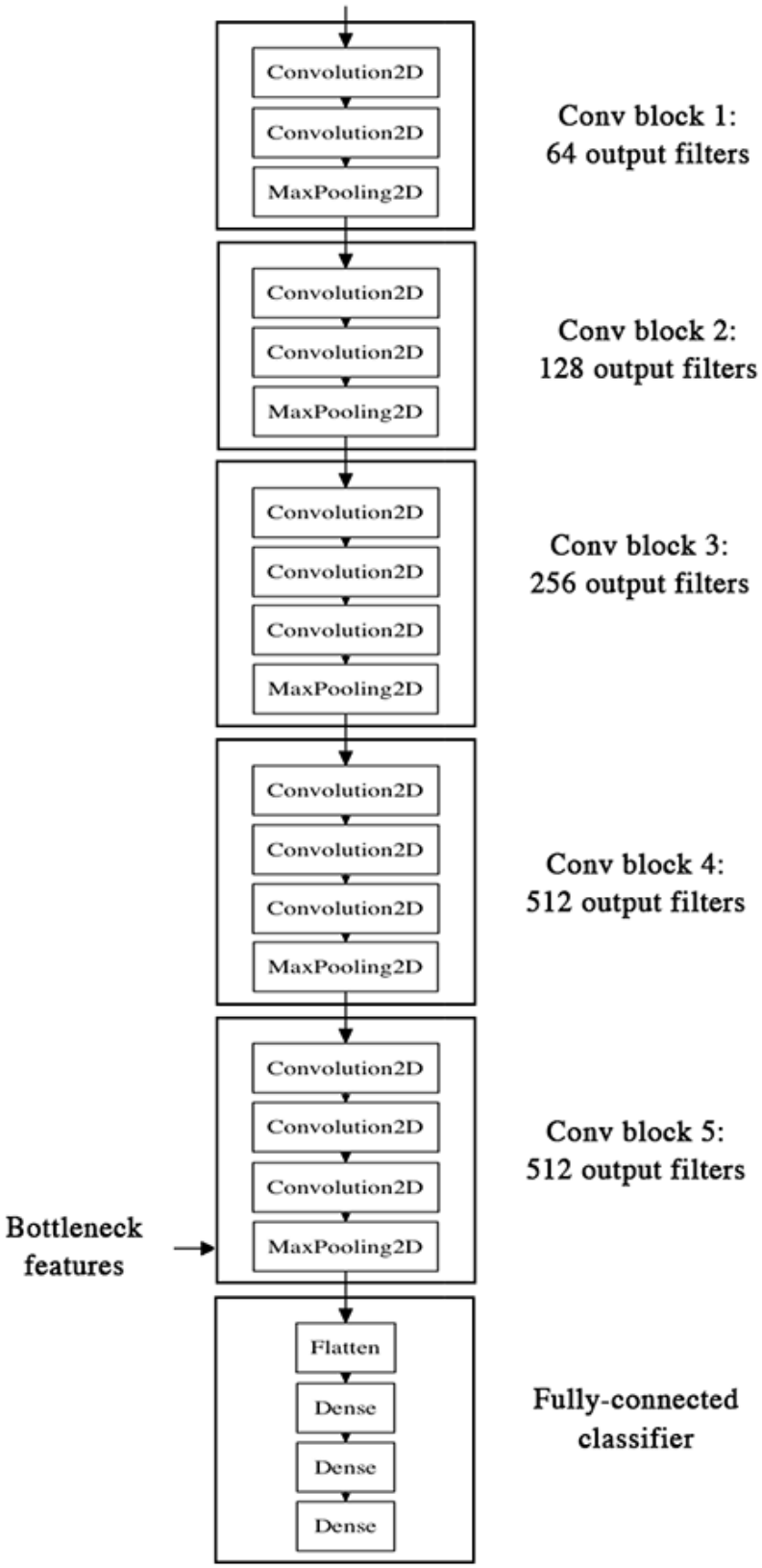

Figure 6: VGG16 architecture

Notice that I here I use LeakyReLU instead of ReLU (Figure 7).

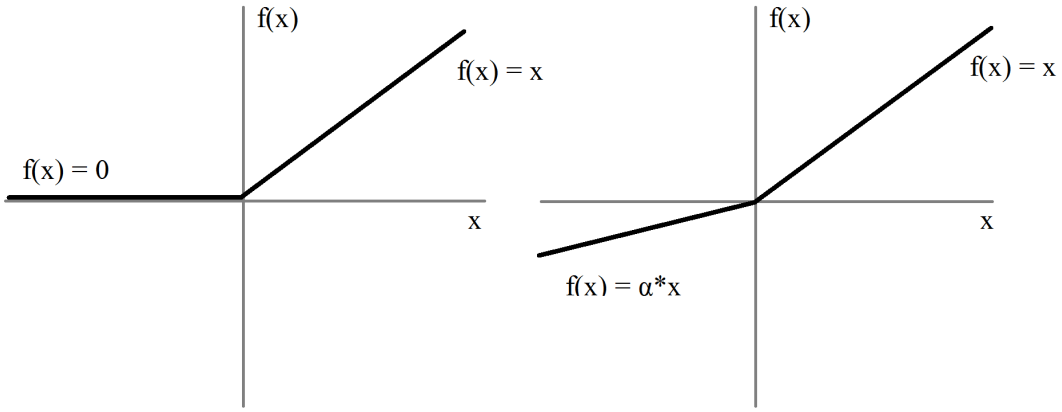

Figure 7: ReLU (left) and LeakyReLU (right)

LeakyReLU is a version of ReLU. LeakyReLU allows a small, non-zero gradient when the unit is not active

Refinement

Optimizer helps the neural network get better weights and achieve better training. Here, I use Stochastic Gradient Descent (SGD). SGD minimizes the loss function by updating the parameters (weights, biases) after each training example or a subset of training examples. SGD has 2 hyperparameters which are learning rate and momentum.

Learning rate determines the size of the steps taken to reach a (local) minimum.

Momentum adds a fraction to the previous weight. It helps increase the size of the steps taken towards the minimum if the gradient keeps pointing in the same direction; and smooth out the variations if the gradient keeps changing direction.

Below is the weight updated function SGD with momentum and learning rate:

$$\nu_t = \gamma \nu_{t-1} + \eta \nabla_\theta J(\theta)$$
$$\theta = \theta + \nu_t$$

ν_t : current velocity vector
ν_{t-1}: previous velocity vector
γ: momentum
η: learning rate
θ: parameter vector
$J(\theta)$: loss function

The training has 2 steps. The network is trained slowly with very low learning rate of 10^{-5} and momentum of 0.2 for the first 8 epochs. Then it is trained faster with higher learning rate of 10^{-4} and momentum of 0.9 for the next 20 epochs.

IV. Results

Model Evaluation and Validation

After trying different deep neural networks as well as manually building a new neural network architecture, VGG16 works the best of all.

VGG16 is a very deep neural network with 5 convolutional blocks

- Block 1 and 2, each of them has 2 convolutional layers followed by a max pooling layer
- Block 3, 4, and 5, each of them has 3 convolutional layers followed by a max pooling layer
- The convolutional layer uses filter of 3x3
    - Block 1 outputs 64 filters
    - Block 2 outputs 128 filters
    - Block 3 outputs 256 filters
    - Blocks 4 and 5, each of which outputs 512 filters
- The max pooling layer uses filter of 2x2 with stride of 2
- The last fully connected layer output an array having length of 5 represented for 5 classes of sea lions.
- The training runs for 28 epochs
- The learning rate and momentum are changed to higher values after the first 8 epochs for faster convergence.

Justification

The trained model is tested using validation set. The graph below shows the loss of training and validation set.

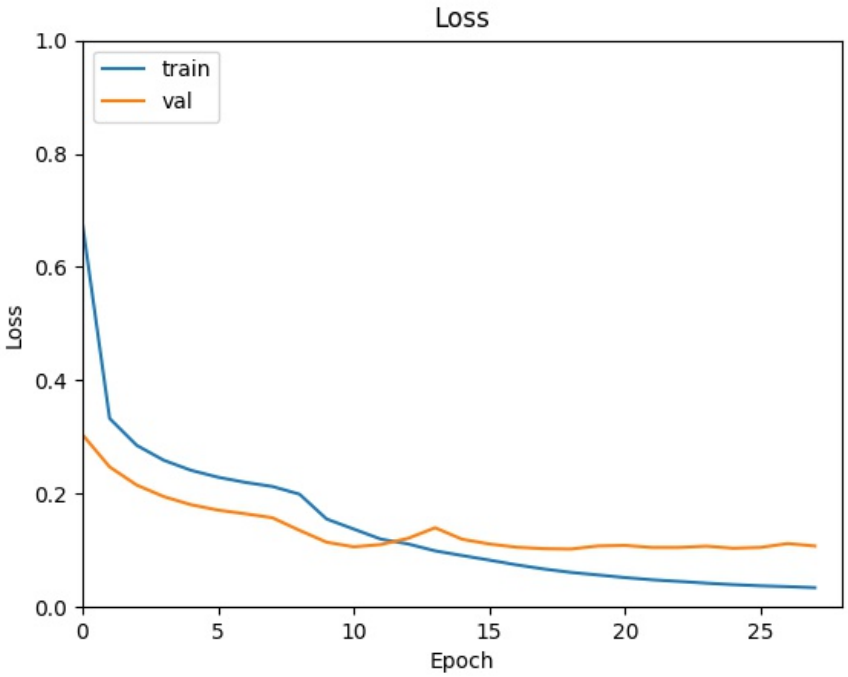

Figure 8: Loss graph of training and validation

The final loss of the model is 10.11% which is a good error rate compared to the number mentioned in Benchmark section.

V. Conclusion

Reflection

The process used for this project is described in following steps

1. A problem was defined.
2. The dataset was downloaded.
3. The data (images) was preprocessed.
4. A benchmark was defined for the classifier
5. A deep neural network was defined and trained using downloaded dataset
6. The model was tested using validation set.

Preprocess the data (3) and finding the suitable neural network to solve the problem (5) are the hardest steps. The data, images, requires multiple steps of preprocessing. The label for the dataset is also got during preprocessing step.

Improvement

To achieve lower error rate, the neural network can be improved by adding:

- Batch Normalization
- Different momentum, learning rate values
- Longer Training
- Larger dataset
- A deeper neural network

The solution for this project can practically help biologists in real world to count sea lions as well as other kinds of animal.

References

1. LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.
2. Parkhi, Omkar M., Andrea Vedaldi, and Andrew Zisserman. "Deep Face Recognition." *BMVC*. Vol. 1. No. 3. 2015.
3. http://cs231n.github.io/
4. http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution
5. https://www.kaggle.com/c/noaa-fisheries-steller-sea-lion-population-count
6. https://keras.io/