Machine Learning Engineer Nanodegree

Capstone Project

Kenta Suzuki

July 14, 2017

I. Definition

Project Overview

Image recognition has gained a lot of attention since 1990s when an algorithm called 'Convolutional Networks' [1]was used to read checks. Accuracy of the networks has been improved over the last few years. For example, an error rate of winning algorithms of ImageNetILSVRC where researchers submit their solutions to the image classification improved from over 26% to over 3% in 5 years.

As "Extended Intelligence"[2], an idea started at MIT Media Lab, suggests, computer vision is one of the interesting areas where machines can extend human capabilities rather than a replacement and working on this field is interesting.

In this project, convolutional networks are used to classify two types of pictures; a picture containing a invasive species and a picture not containing the species.

Problem Statement

Computer vision can be extended to natural environment. For example, there are many species that have harmful effects on the ecosystem. Currently, trained scientists visit some areas and

check the existence of these harmful species. They would be able to concentrate on the research where their creativity and tacit knowledge are required if a reliable algorithm automates this classification process.

Thus the purpose of this project is to develop an algorithm which classifies the images with low error rate in order to reduce the burden imposed on the scientists.

Convolutional Neural Networks are used for addressing this computer vision problem. Model with different parameters and architectures are trained on the training data and final accuracy is reported on based on the test data. An anticipated solution is an ensemble method, a combination of many individual networks. ResNet[3], which won the ImageNetILSVRC 2015 with error rate under 3.6%, might be a model with the highest accuracy.

Metrics

Metrics used are as follows:

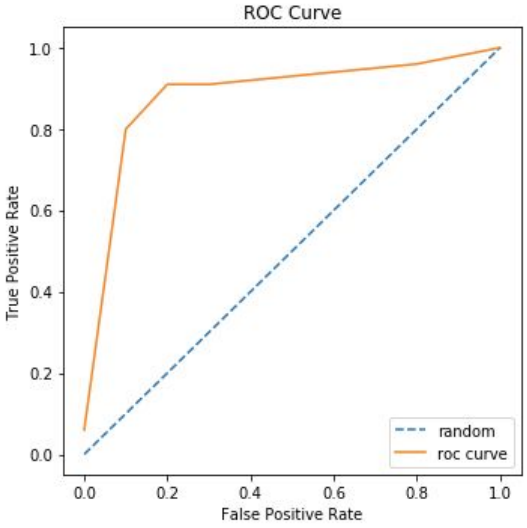
1) Binary accuracy: the proportion of samples an algorithm correctly classifies. This is a useful metric not only for measuring the how many sample are correctly classified but for the detection of overfitting which often occurs when small data sets or complex algorithms are used.

2) Binary cross-entropy: loss function to minimize. This is useful for deciding an appropriate learning rate of optimization.

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

Figure 1: Formula of binary cross-entropy

3) AUC: This is useful especially when the data sets are imbalanced such as many samples are drawn from one class and few from the other class. Predicting every sample as a major class will result in high accuracy rate in such data sets and a trained algorithm would not classify the minority class correctly. AUC is a measure of tension between true positive rate and false positive rate. As an example, if data is distributed as [0, 1, 1, 1, 1, 1, 1, 1, 1, 0](0 and 1 indicate each label), always predicting label 1 results in 80% of accuracy. However, the false positive rate (formula is shown below) is 1.0. This could have negative impact. In this project, this could mean that some actions will be taken against places which are misclassified to have invasive species. Reducing false positive rate will reduce this cost, and AUC is a good measure for that.



$$FPR = \frac{FP}{TR + FP}$$

Figure2: ROC curve and formula for false positive rate

II. Analysis

Data Exploration

The datasets are pictures (RGB format) collected in a Brazilian national forest. Some of the images contain invasive species, *Hydrangea*, and the objective is to predict the presence of this species. This datasets are publically available on Kaggle competition [4].

Training data contains 2295 images, testing data contains 1531 images. Labels for the training data are given but not available for testing data.

Data sets are imbalanced. There are 847 images with no Hydrangea, and 1448 images with Hydrangea appearing.

Figure 3. (left) No presence of Hydrangea

(right) Presence of Hydrangea

Exploratory Visualization

To understand the class distribution visually, bar chart for sample proportion for each class is plotted below.

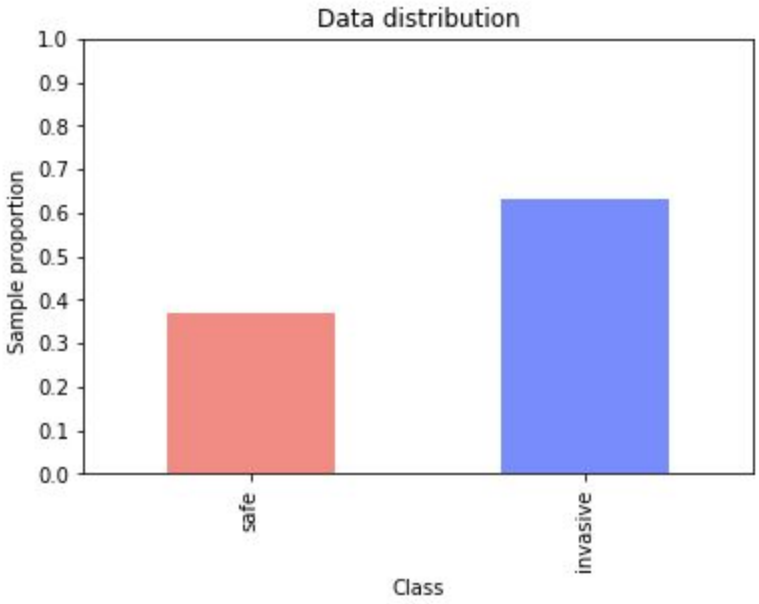


Figure 4: Sample proportion for each class. Safe means the image with non-invasive species.

Ratio: Safe (non-invasive): Invasive = 3.5 : 6.7

Algorithms and Techniques

Convolutional neural networks are used for this problem. They have a main advantage over fully connected neural networks; They retains the spatial information of the image while fully connected networks squishes pixels into 1d vector. The main difference is the use of kernel filters and pooling layers.

1) Kernel filters: These are applied to the local regions in the input and filter size determines the depth of output layers. Each layer of depth learns different patterns of an image.
2) Pooling layers: These reduces the number of parameters by downsizing the input volume. This leads to the computational efficiency and the networks also becomes robust to overfitting.

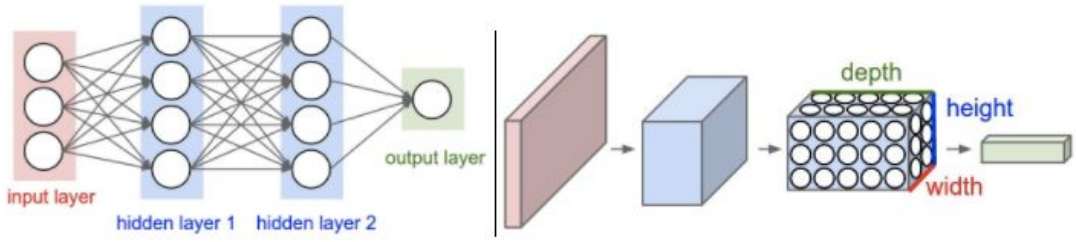


Figure 5: images of fully-connected networks and convolutional neural networks, from CS231n, Stanford [5]

Possible parameters to consider are:

1) Activation function (this function is applied after weights and inputs are multiplied. Some functions alleviate gradient vanishing problem, thus resulting in faster convergence)
2) Optimizer (this is used to minimize the error. Different optimizers result in various error rate and convergence time)
3) Depth of net architecture (this makes the networks complex to learn more different patterns of images. As a side effect, this increases the number of parameters, thus increasing the chance of overfitting)

4) Weight decay (this prevents overfitting but strong decay might lead to underfitting)
5) Number of epochs (the number of trainings)
6) Batch size (the number of samples on which the weights are updated for each epoch. This determines the variance and computation time)

Benchmark

As a benchmark, simple convolutional networks with six layers are used. Neither data augmentation nor class weight for tackling the imbalance problem are used. This model is used for a first attempt since it measures the impact of imbalance data within reasonable time.

The AUC score for this model is 0.817. Reliability of other models are compared to this score. Final training loss is 0.029, validation loss is 0.61, training accuracy is 0.99, validation accuracy is 0.82. This model is overfitting on the data, and a discrepancy between AUC and training accuracy indicates that the model suffers from the class imbalance. This might be that data sets are small so that it captures the idiosyncrasies of training data.

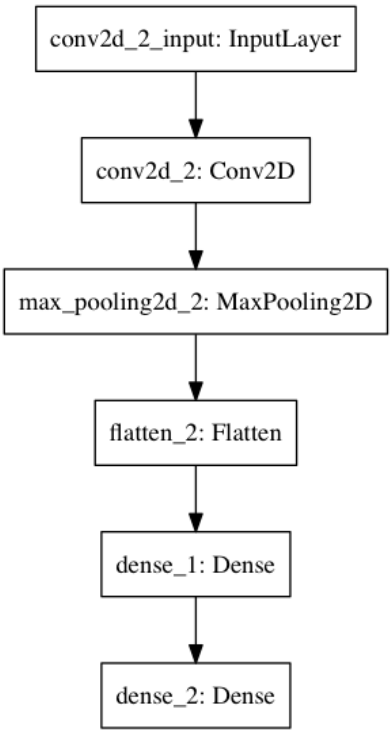


Figure 6: Architecture of convolutional networks as a benchmark

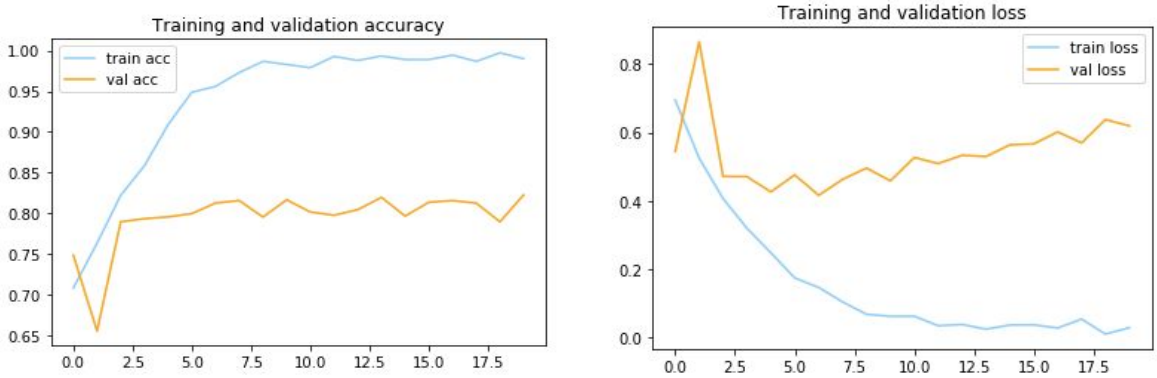


Figure 7: Training, validation accuracy (left)

Training, validation loss (right)

III. Methodology

Data Preprocessing

Each batch samples are normalized by setting their mean to 0 and divide them by its standard deviation so that it ranges from 0 to 1.

For addressing the class imbalance, data augmentation and class weights are used. Since large data sets are required for convolutional networks to work properly, undersampling is not conducted.

Weights are initialized from the gaussian distribution with mean 0 and with standard deviation square root of 2 over the number of units, as discussed in He et al [6].

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \times \sqrt{\frac{2}{\#of\,inputs}}$$

Figure 8: He initialization

Implementation

All models are implemented by Keras as Tensorflow backend. Transfer learning is used for Pre-trained models on ImageNet. Due to the absence of GPU, this method is used as a feature extraction.

Many models are implemented similarly to the convolutional neural networks below.

Algorithm 1: Convolutional Neural Networks

```
Data: train_data, validation_data
1 begin:
2       for i=1 to number-of-epochs:
3           for j=1 to step-size(number-of-training-data//batch_size):
4               extract batch from training data
5               feed them to networks and train the model
6           for k=1 to validation-step(number-of-validation data // batch_size):
7               compute the accuracy and error rate
8           report  mean of the accuracy and error rate
9           if  validation error does not reduce over some epochs:
10              decrease the learning rate
11  end:
```

Figure 9: Procedure for training Convolutional Neural Networks

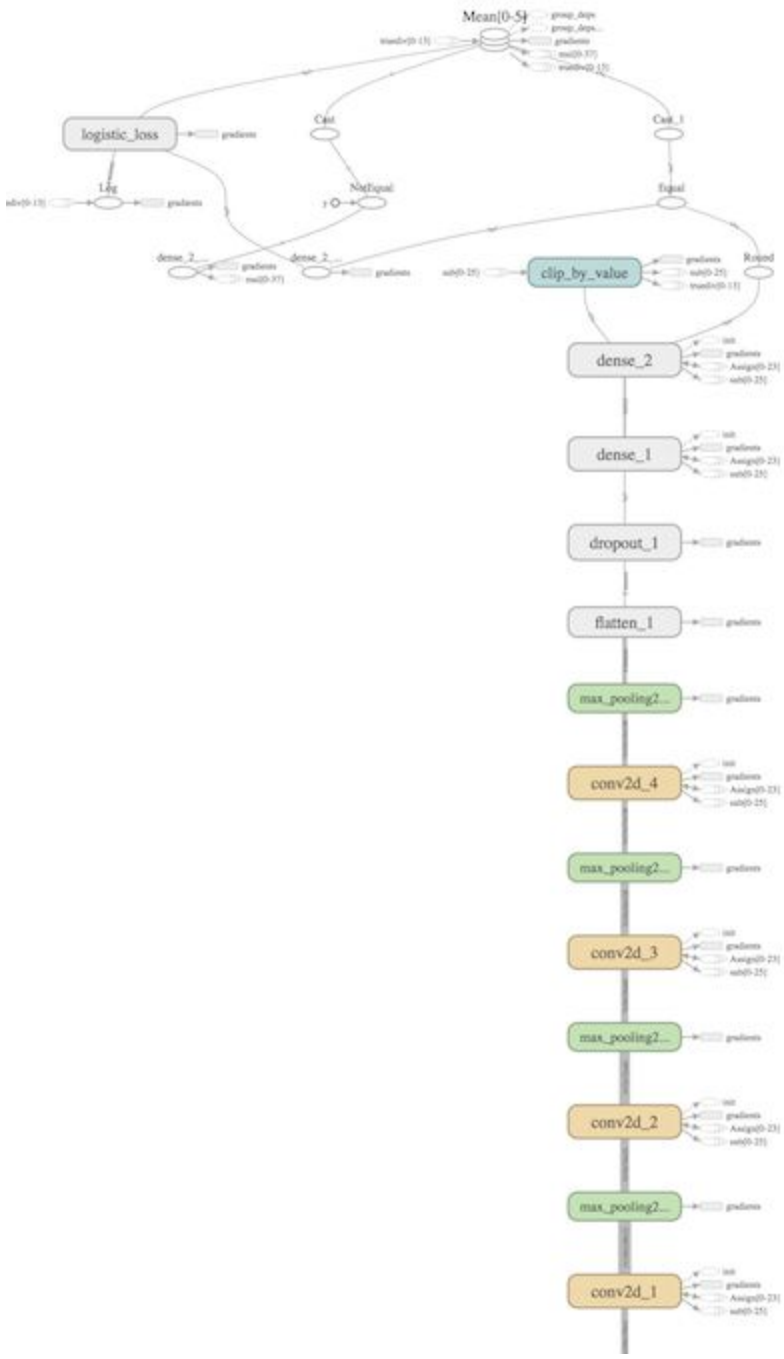


Figure 10: Graph of computational graph.

Its network architectures are composed of :

1) Convolutional layers
2) Max pooling layer
3) Dense layer

Loss is computed using binary-crossentorpy.

Refinement

In order to improve the accuracy of the benchmark, following techniques are added.

1. The depth of net architecture: The main reason behind the benchmark's score is considered to be its depths of networks. In order for the networks to learn different patterns of images, depth is increased.
2. Dynamic learning rate: In order to get out of plateau when minimizing the cost function, learning rate is divided by 10 when validation loss is not reduced for certain periods such as 2 successive epochs.
3. Dropout: As seen in the Figure 7(left), overfitting occurs when training a benchmark. Adding a dropout layer prevents this.
4. Batch normalization: This solves, internal covariate shift and make the convergence faster. Activation function, elu, has same effect.

5. Different optimization: In the benchmark model., RMSProp is used. For comparison, another adaptive learning rate method, Adam is used. These optimizers have the benefit of alleviating the need for tuning the learning rate since the default learning rate often works well with a diminishing learning rate [7].

As seen in the figure 11, a different activation with a different optimizer results in 1-3% different AUC score.

	AUC: Conv2d	val_acc	val_loss
relu: RMSProp	0.90499	0.8951	0.2762
relu: Adam	0.87503	0.9152	0.2044
relu: Adam + Batch-norm	0.89781	0.8454	0.3604
relu: RMSprop + Batch-norm	0.89707	0.8150	0.3791
elu: RMSProp	0.89670	0.8101	0.3792
elu: Adam	0.89970	0.8348	0.3730

Figure 11: AUC, validation accuracy and validation loss for convolutional neural networks. 'Batch-norm' represents 'batch normalization'.

IV. Results

Model Evaluation and Validation

During training a classifier, a validation data is used to monitor the accuracy and loss. Parameters are chosen based on AUC score of Convolutional Neural Networks with 13 layers. These parameter are also applied to the dense layers on top of pre-trained models.
The final model is an ensemble with weighted average of the seven models that result in top AUC score. The training process for the final model is as follows:

1. The number of epochs, 20: the data set is small and validation accuracy and validation loss do not improve within this training steps
2. The learning rate is divided by 10 when validation loss is not decreased for 2 epochs.
3. Class weights: this penalizes a model when it misclassifies the minority class

The robustness of the final model is measured by AUC scores produced by the best single classifier, VGG16, for computational efficiency. It is compared among training, validation ,testing data and additional 19 images obtained from the Internet. As seen in the Figure 12 below, AUC score for training, validation, test and new images are similar. This indicates the robustness of the model.

	new	test	train	validation
AUC	0.954	0.95192	0.98122	0.953552

Figure 12: AUC score for each data sets

Justification

AUC score of the ensemble model is 0.958. This is 16.8 % improvement than the score produced by benchmark model. The seven models used are: VGG16 with dropout rate 0.5, VGG16 with dropout rate 0.7, Convolutional Networks with 13 layers, Resnet, VGG19, InceptionV3, Xception. Models except the convnets with 13 layers are trained on imagenet and applied to this binary classification as transfer learning. The AUC score is significant enough to solve the objective; it gives a score better than a random predictor does and it minimizes the false positive rate. This model can be used to find the image of Hydrangea. However, it will not be generalized to detecting general invasive species besides Hydrangea, which is crucial since there are other invasive species and classifying them is important.

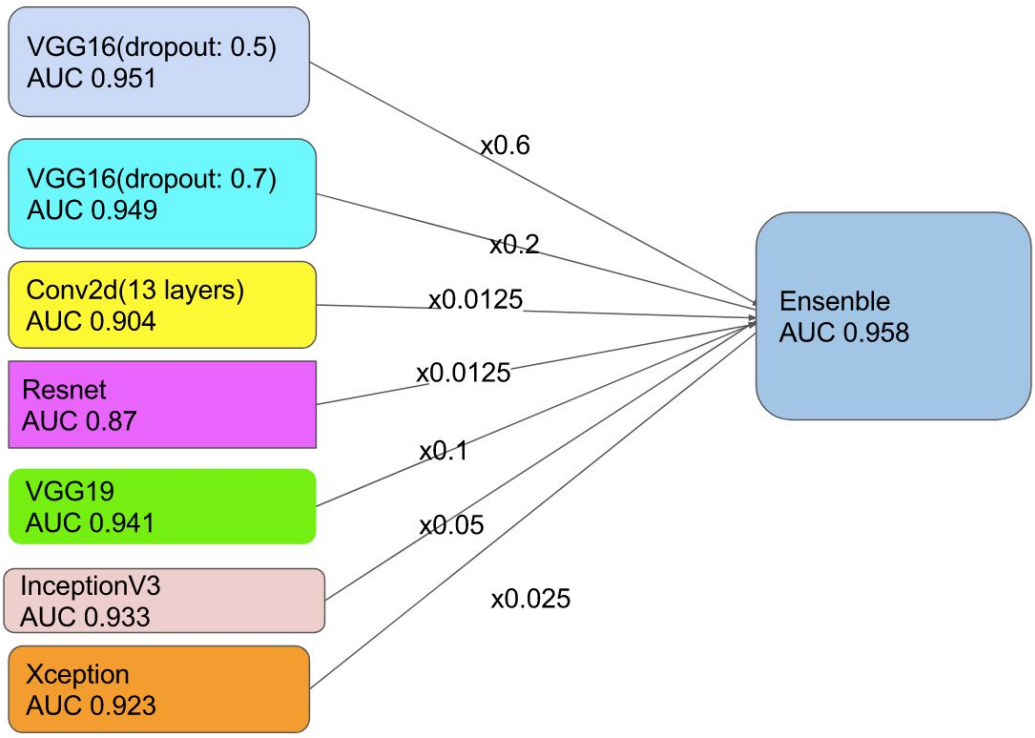


Figure 13: Final model, ensemble model of 7 models. Each model is weighted for its importance.

V. Conclusion

Visualization

There are 21 false positives in the validation data. The images a model predicted with 98% confidence are presented below.

Figure 12: images of non-invasive species but classified as the ones of invasive

These images contain some flowers, which seems to confuse the model for an invasive species.

Visualization of filters of convolutional neural networks makes it possible to see which layer learns what characteristics of an input image. For visualization, convolutional neural networks with 5 convolutional layers are used. Below are images for what each 5 convolutional layer recognizes the image of invasive species. Although there is no clear pattern for each layer, first layers seems to detect tiny edge patterns and last layer sees more abstract image.

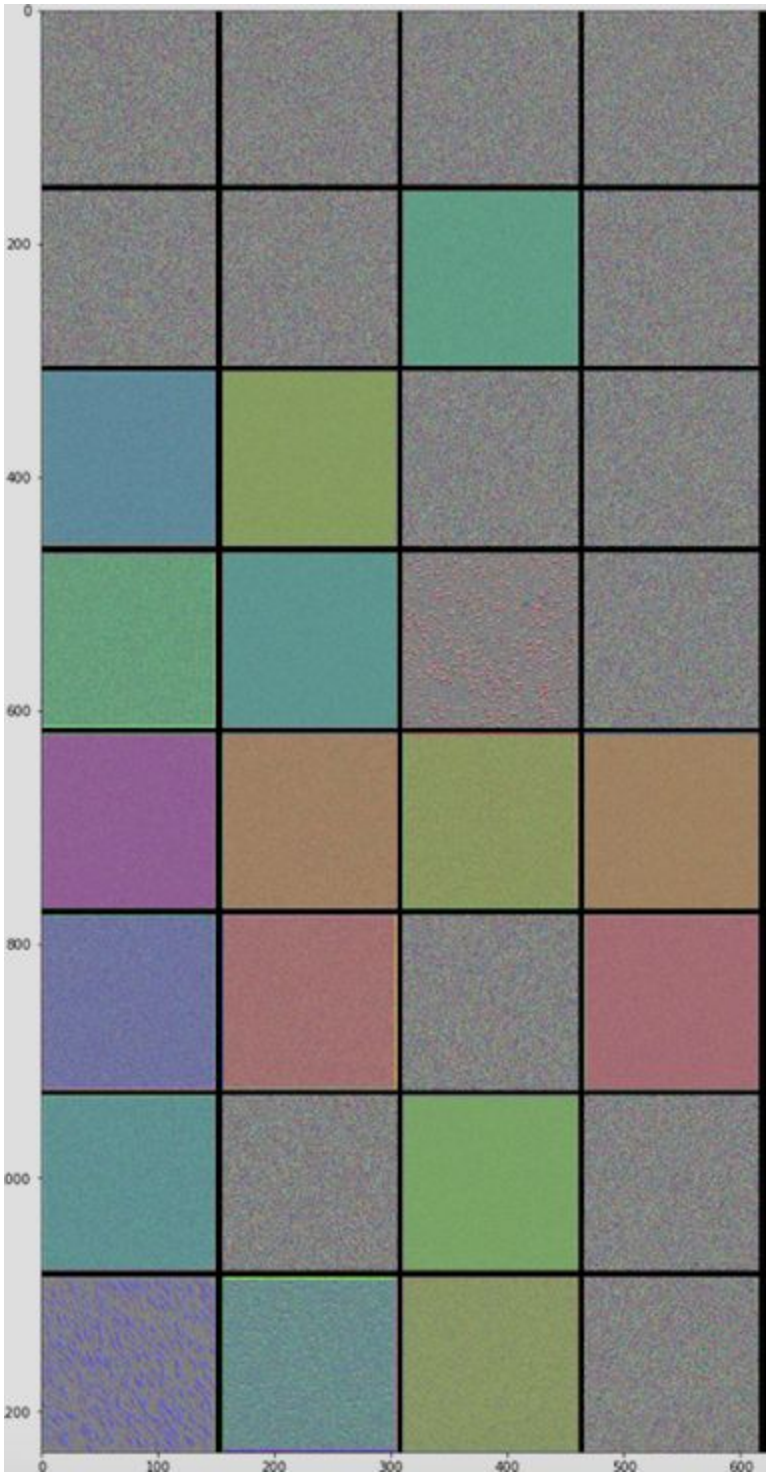


Figure 13 : Filter patterns for convolutional layer 1

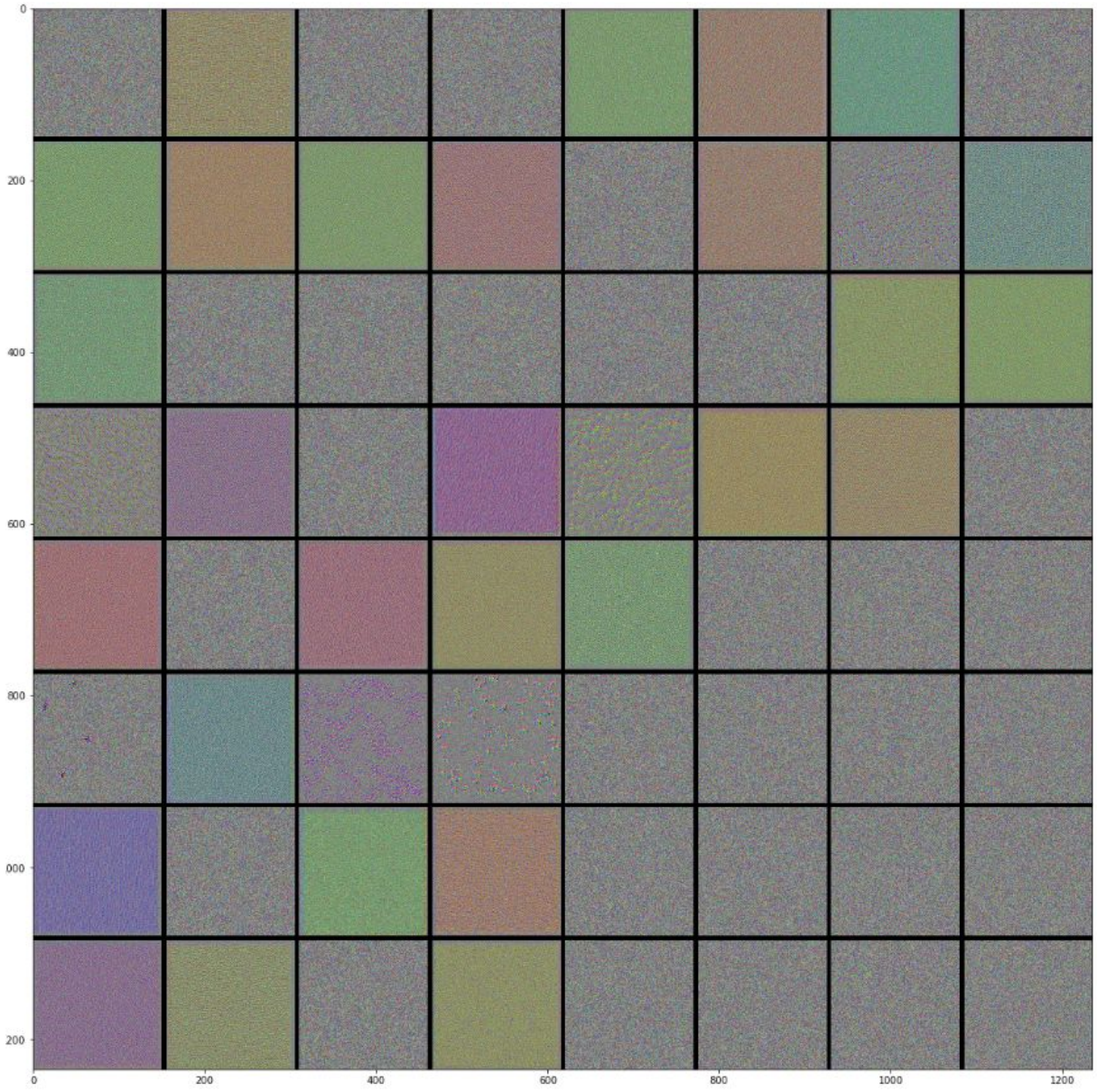


Fig 14: Filter patterns for convolutional layer 2

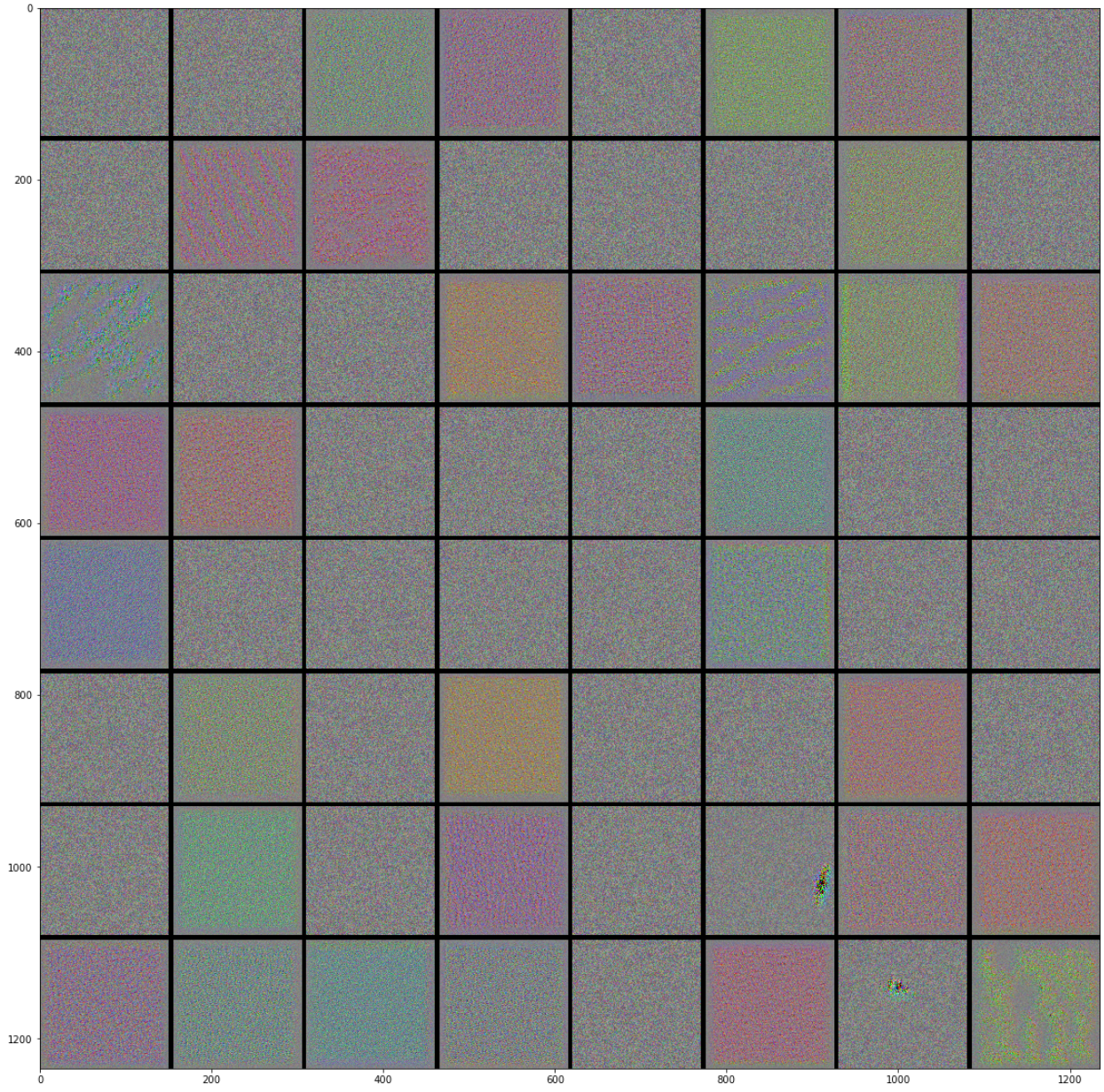


Fig 15: Filter patterns for convolutional layer 3

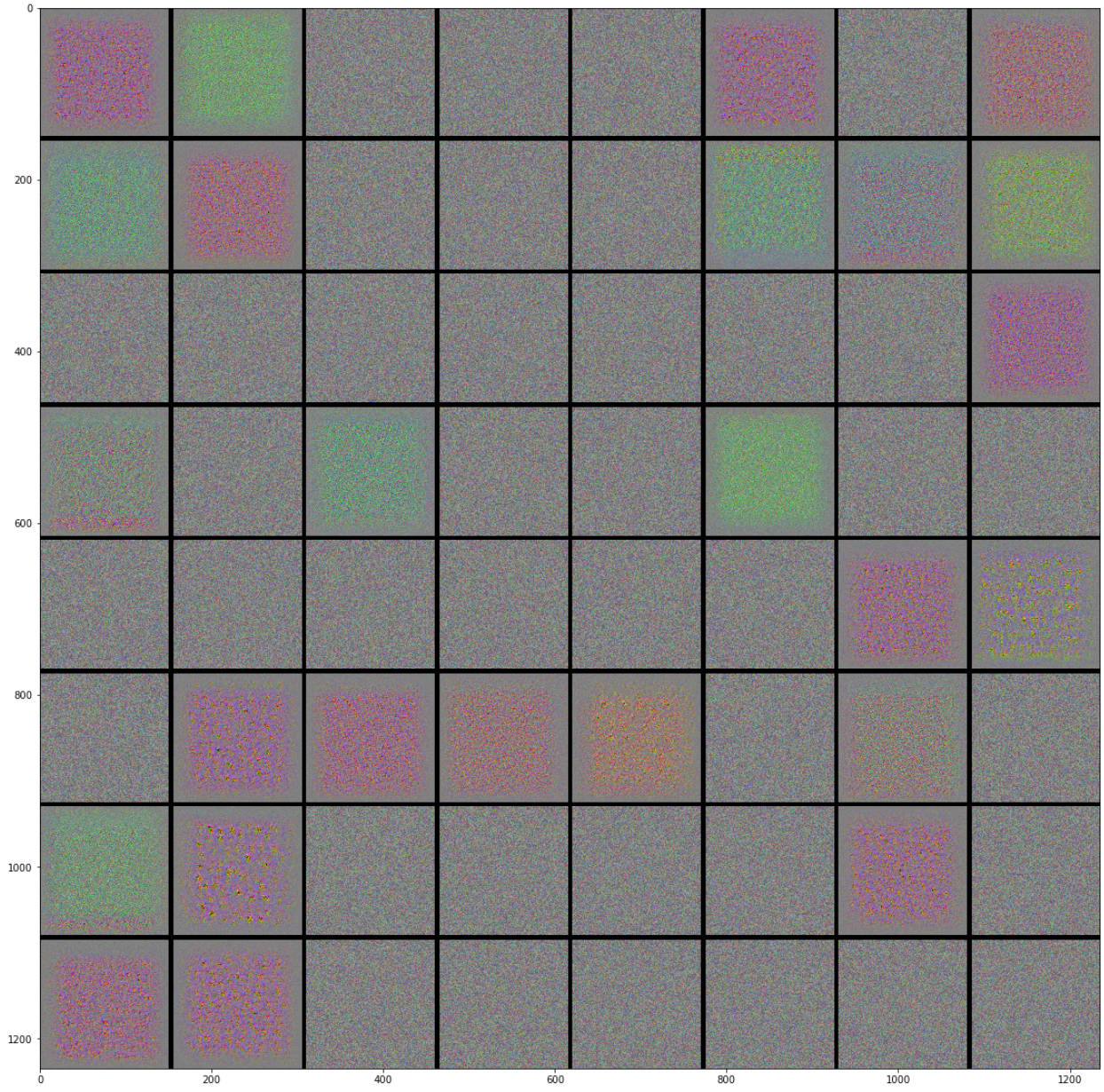


Fig 16: Filter patterns for convolutional layer 4

Reflection

The procedure for the the final solution to this problem is summarized as follows:

1. The dataset is divided into training and validation data
2. The data is preprocessed to make it to be an appropriate format for being fed into the neural networks
3. A benchmark model is trained and its accuracy is reported
4. The classifier is trained with a different combination of parameters
5. Pretrained models are used as features extraction and add another layer on top (transfer-learning)
6. Ensemble method using top 7 individual models is used to produce the final score

Step 4 and Step 5 are challenging parts because of the computation time. Training classifiers without GPUs took several hours, and random search for finding a good combination of parameters was impossible.

Since the size of data sets is small , increasing the number of epochs did not seem to increase AUC.

Improvement

Since all models were trained without GPUs, training convolutional base, fine-tuning and random search for hyper-parameter optimization were impossible given its computational expensiveness. Since pre-trained models are tailored to the images of ImageNet, modifying the architectures of the networks will improve AUC score. Collecting more data from other sources might improve the accuracy especially for ResNet since data size is too small for its number of complex architecture with many parameters.

References:

[1] LeCun.Y. LeNet-5 - Yann LeCun http://yann.lecun.com/exdb/lenet/

[2] Ito.J. Extended Intelligence https://www.pubpub.org/pub/extended-intelligence, 2016

[3]He.K. , Zhang.X. , Ren.S. , Sun.J, Deep Residual Learning for Image Recognition https://arxiv.org/abs/1512.03385

[4] Invasive Species Monitoring | Kaggle https://www.kaggle.com/c/invasive-species-monitoring/data, 2017

[5] CS231n, Convolutional Neural Networks: Architectures, Convolution / Pooling Layers http://cs231n.github.io/convolutional-networks/

[6] He.K. , Zhang.X. , Ren.S. , Sun.J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification https://arxiv.org/abs/1502.01852, 2015

[7] Ruder.S., An overview of gradient descent optimization algorithms http://ruder.io/optimizing-gradient-descent/index.html#adam