Classifiying Transmission Electron Microscopy Virus Textures

Alan Do-Omri 260532985
Kian Kenyon-Dean 260564475
Genevieve Fried 260564432

December 14, 2015

Abstract

On a daily basis doctors are tasked with determining the identity of viruses from blood samples. Even with advanced microscopy techniques, viral strains can be difficult to discern with the human eye. Misclassification of viruses can have grave implications, not only for the patients in question but also for those who come into contact with him or her, so it is a critical problem demanding high accuracy. Fortunately, virus classification can be an image classification problem, and machine learning methods present exceedingly high accuracy rates for image classification tasks without requiring expert knowledge to do so. Here we present a successful application of Convolutional Neural Networks (CNN) to the problem of virus classification. Using the Virus Texture Dataset from Uppsala University, we classify viruses samples into one of 15 categories and compare its performance to previous work done on the same task.

1. Introduction

Identifying viral pathogens, existing and emerging, is a critical priority for public health. Current diagnostic methods include direct examination of a speciman by an trained medical professional using Electron Microscopy methods. Unfortunately, even with magnification techniques it can be difficult for doctors to identify specific viral types.

On the other hand, automatic classification methods in machine learning have proven quite successful at image classification. In particular, research employing neural networks has reshaped image classification in recent years Schmidhuber [12]. Specifically, Max Pooling Convolutional Neural Networks have set the bar at it's highest for an array of tasks, including the multi-digit number recognition in Google Street View images, reCAPTCHA box classification Goodfellow et al. [6], and ImageNet classification Krizhevsky et al. [7].

The high degree of accuracy obtained with these automated methods is highly desirable for a problem like virus classification. We utilize state-of-the-art machine learning methods for image processing by treating virus classification as an image classification problem. Our dataset is the Virus Texture Dataset from Uppsala University that contains images of fifteen different virus types; we extend the dataset with random rotations of the images.

We experiment with an array of machine learning classifiers, including Support Vector Machines with linear and radial basis function kernels, Gaussian Naive Bayes, Logistic Regression, Random Forest Trees and Feed Forward Neural Networks with local binary pattern feature extractors. Our best results come from our Convolutional Neural Network (CNN) using max pooling, with which we classify virus samples from our dataset into one of 15 categories with an accuracy of approximately 85%.

2. Problem Definition

Our research takes place in a greater context of a desire to use machine learning classification meth-

ods to accurately classify viruses from viral sample images. This extends to both viruses which are currently known, and potentially evolved viruses like the H1N1 influenza pandemic virus in Mexico and the US in 2009, which was a modified version of known influenza strains. Convolutional Neural Networks with all the function approximation power of Neural Networks, but specifically architected for images, present a highly promising solution for computer automated virus identification, and a useful supplementation to traditional virology research.

From our survey of the work done in virus classification, treating virus classification as an image classification problem is not common. Most researchers have used Support Vector Machines, tree modeling structures like Random Forests, and Feed Forward Neural Networks to classify and predict known and emerging viral strains through analysis of genome segments and genetic sequences Raj et al. [11] Attaluri et al. [1] Attaluri et al. [2]. Kylberg et al. [9]'s paper, which we looked to for insight into feature extraction methods, uses a Random Forest classifier to assess the discriminant potential of various texture measures on virus images. As far as we can tell, using a CNN is rather unprecedented for this problem. Thus we find the application of CNNs to virus classification a pertinent one that would be fruitful to explore further.

3. Related Work

In the work of Kylberg et al. [9], texture analysis is performed on 15 different virus samples and the resulting feature vector is fed into a Random Forest classifier. The authors first compare the performance of different texture analysers. These texture analysers include Local Binary Patterns (LBP), Radial Density Profile (RDP), and respective variants of the two.

We implement the LBP feature extraction method in our Feed Forward Neural Networks and Random Forest Classifiers because a variant of it performs best among the texture measures used on virus images in Kylberg et al. [9]. With respect to Kylberg et al. [9] results, the first thing to note is the differences betwen ours and their preprocessing methods. Kylberg et al. [9] made modifications to the scales of their images. In one case, one pixel in the image corresponds to one nanometer, which they refer to as *fixed scale*. In the other case, the radius of the virus is set to take twenty pixels, and they refer to this as *object scale*.

They implement a Random Forest classifier with 100 trees (we do the same for comparative purposes) and their results can be found in figure 1. The results of six different texture extractors are shown in this figure, but our area of interest is the first row of results for LBP^{ri}. "ri" stands for rotational invariant and is a way to reduce the size of a feature vector by rotating a vector bitwise to get the smallest possible number (for e.g. the number 110 becomes 011).

4. Methodology

4.1. Feature Design and Selection Methods

We emulate one of the feature extraction methods found in Kylberg et al. [9], Local Binary Patterns (LBP). LBPs work as follows:

Given an image, for each pixel p_i in it we sample n equally-spaced points on the circle of radius r with center p_i – an example of sampling is shown in figure 3 – and construct a vector $v(p_i)$ such that its ith entry is a 1 if the ith sampled pixel has a value bigger than p_i, and 0 otherwise.

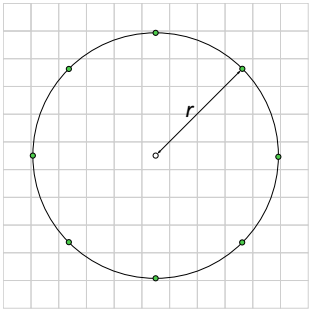


Figure 2: Example of LBP sampling: The green points are the neighbouring sample points at distance r from the central white point. In this case, we are sampling with $n = 8$ neighbour points.

A sequence of 0s and 1s are constructed $v(p_i)$ to form a binary number v_{p_i}. Once we have the v_{p_i} for all pixels, we construct a histogram counting the number of appearances of each value v_{p_i}. The histogram forms the feature vector associated with the given image. Kylberg et al. [9] denote this feature extraction method by $\text{LBP}_{n,r}$, where n is the number of sampled points and r is the radius, as described above. The resulting histogram is represented in a vector of counts with 2^n elements.

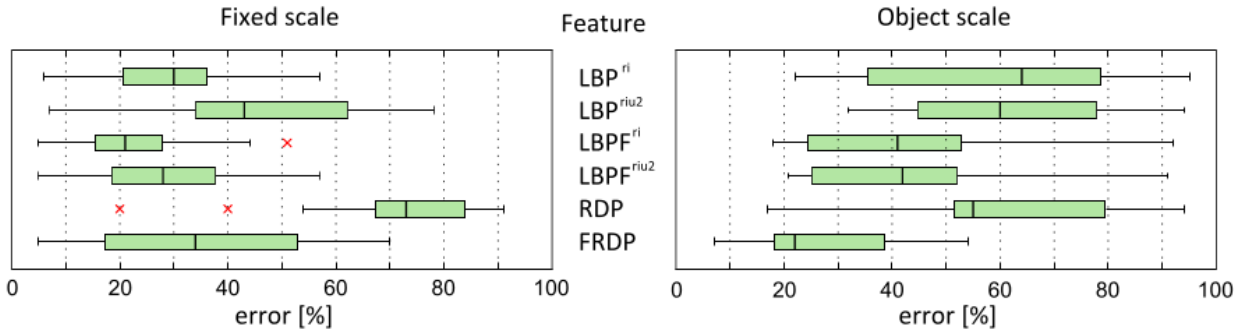


Figure 1: This figure comes from Kylberg et al. [9]. The classification errors are shown for a Random Forest classifier using the 6 different texture extractors as described in section 3. The boxes' vertical lines represent the median and the red × represent outliers that are at least 1.5 times the size of the box away from it. The error bars are from the lower to the upper quartile. We're only concerned with the first row of results for LBP^{ri}.

4.2. Dataset and Pre-Processing Methods

The Virus Texture Dataset contains images (texture samples) of 15 virus types obtained through transmission electron microscopy (TEM). The texture samples are extracted using an automatic segmentation method used in Kylberg et al. [8] that detects virus particles of various shapes in TEM using a series of analytical steps. Each virus class has 100 unique texture patches of size 41x41 pixels.

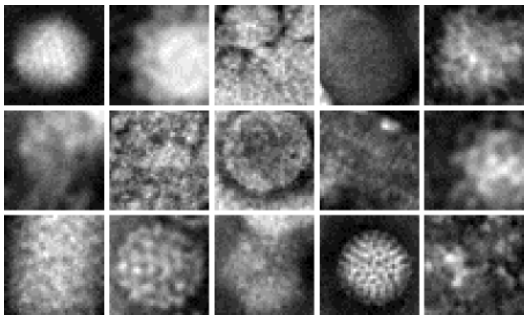


Figure 3: Example images from the Virus Texture Dataset, created by Kylberg et al. [9].

The dataset is small for the purposes of improving our generalisation ability; we extend it by generating twelve random rotations for each image in the dataset, producing 18,000 more images.

5. Baseline Classification Approach

We tested Support Vector Machines (linear and radial basis kernels (rbf)), Logistic Regression, and Gaussian Naive Bayes using the scikit-learn Pedregosa et al. [10] machine learning library. We use these classifiers as a baseline to provide insight into our datasets complexity. We trained and tested these classifiers over the rotated dataset. SVM with an rbf kernel performs the best since it allows for projection of the pixel feature space to a higher dimension, thus making it capable of capturing the non linearity of our data in a way that the other classifiers requiring linearly seperable data cannot. These classifiers clearly have poor results because none of them are specifically architected to handle images. See the appendix for the confusion matrices of the results.

Classifier	Parameters	F1-Accuracy
Logistic Regression	default	0.271
Linear SVM	linear kernel	0.248
SVM	rbf kernel	0.327
Gaussian Naive Bayes	none	0.273

6. Random Forest Approach

We tested several different Random Forest configurations when using LBP feature extraction on the rotated dataset. The results, while better than the

baseline classifiers above, still have room for improvement. We obtained 43% accuracy with 90 decision trees in the forest, which is less than what is obtained by Kylberg et al. [9], who investigate more feature extraction methods than we do.

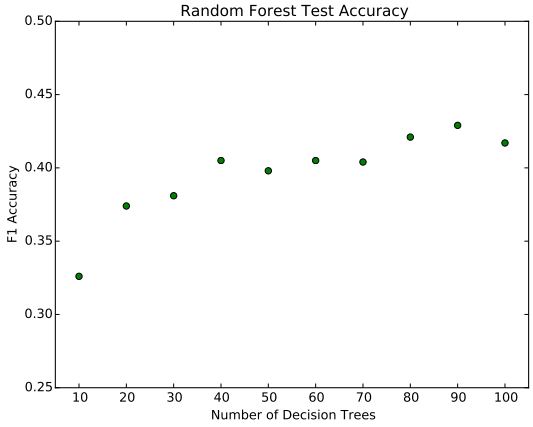


Figure 4: Test-set results from random forest classification with default parameters.

Since our focus is not on Random Forest classification, and our results were far from optimal, we figured that futher hyperparameter optimization would not result in significant improvement. Implementing more complex cariants of LBP as described in Kylberg et al. [9] could improve our Random Forest classifier's results, although to truly mimic Kylberg et al. [9]'s results we would have had to use 16-bit images as opposed to 8-bit images.

7. Neural Networks Approach

We take inspiration from Kylberg et al. [9] again and use LBP features in a Feed Forward Neural Network (FFNN). Our FFNN is built using the Lasagne and Theano Bastien et al. [3], Bergstra et al. [4] libraries. In our experiments, we used $\text{LBP}_{8,2}$, the default implementation of the *Mahotas* computer vision library Coelho [5]. The values of $8, 2$ come from Kylberg et al. [9], who achieve their best results with these parameters. *Mahotas*' implementation of LBP is such that the feature vector is of smaller dimension than the usual dimension of a histogram with 2^N values where N is the number of sampled points.

7.1. Results

To normalize the data, we chose to divide the entire dataset by 1.1 times the maximum value of a feature in the training set in order to be certain that no features in the normalized test set had a value greater than 1. In all cases, we use a learning rate of 0.005 and an L2 regularization weight of 0.0001, both arbitrarily chosen, and a 15 units softmax layer to conclude our network. The dataset described in section 4.2 is shuffled, and the training is done with a batch size of 16 by stochastic gradient descent. All images pertaining to the results, such as the confusion matrices for the validation and testing set and the learning curves, can be found in the Appendix: the first neural network results are in figures 6 and 7; the second neural network results are in figures 8 and 9; the third nerual network results are found in figures 10 and 11. The table below shows the results obtained on the test set. The first column represent, in order, the number of units in each of the hidden layers.

Hidden Layers	Test Error
256 units	52.77 %
256 and 256 units	52.22 %
256, 128 and 64 units	51.94 %

7.2. Discussion

We notice that our neural network with one hidden layer took more time to converge than our three hidden layer network. The network with one hidden layer also had the largest error, and we think this is because the network did not have enough neurons to learn properly. Meanwhile, our network with three hidden layers was quick to overfit. This is to be expected since it had many more parameters to learn than the network with one layer. Nevertheless, our three hidden layer network gave the best results, and could be improved further by adding Gaussian noise to its neurons and applying dropout.

Compared to the work of Kylberg et al. [9], we can see from Figure 1 that the LBP^{ri} on fixed scale performed at about 30 % error. whereas on the object scale it performed at about 63 %. Since we didn't perform any rescaling of the images, we expect the error to be within that interval. On the other hand, resizing to fixed scale would certainly improve the accuracy. Finally, *Mahotas*' implementation of LBP does not necessarily match the implementation of LBP^{ri} so this can induce some error as well.

8. Convolutional Neural Network Approach

CNNs are often used in image classification tasks as they possess the power of Feed Forward Neural Networks but are specifically structured to process images efficiently. As we approached the problem of virus classification from an image classification front, CNNs seemed a natural method to use. Our CNN is built with the Lasagne and Theano Bastien et al. [3], Bergstra et al. [4] libraries. Its first convolutional layer is fed by a 41x41 normalized input image. This layer consists of 16 8x8 filters, followed by a 2x2 max pooling layer. Two more convolutions are performed on top of this, the first with 48 5x5 filters, the second with 60 2x2 filters; both convolutions are fed into a 2x2 max pooling layer. Two fully connected layers with 50% dropout on their inputs conclude the network. We used a learning rate of 0.005 and an L2 regularization weight of 0.0001, both arbitrarily chosen. Normalization was performed to place all pixel intensities between a range of 0 and 255. Training was performed with a batch size of 60 by stochastic gradient descent.

8.1. Results

Images pertaining to the results of our CNN can be found in figures 12 and 13 in the Appendix. The table below shows the best error costs found on our training, validation and test sets.

Best Train Error	0.0120 %
Best Validation Error	0.1500 %
Best Test Error	0.1511 %

8.2. Discussion

Our CNN classifies viruses on our training set with an accuracy of 85%. Given that we passed in raw images to the network and still achieve nearly as high accuracy results as Kylberg et al. [9] highly optimized Random Forest, which obtained 87%, it is clear that a highly optimized CNN would surpass their results for this problem. Our results might have been improved with a more exhaustive hyperparameter search. The search itself is computationally expensive as CNNs, and Neural Networks in general, have many parameters. Nevertheless, experimenting with the number of convolutional layers, percentage of dropout, learning rate etc seems a worthwhile investment.

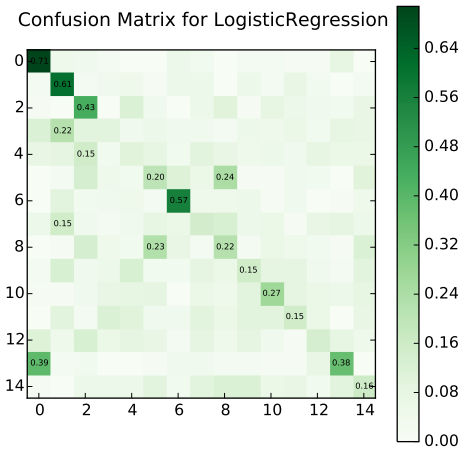
From an medical perpsective, even 87% accuracy is not high enough accuracy for the critical nature of this task; accuracy close to 100% is desired, and an optimized CNN might achieve this. As it currently stands, however, this CNN could certainly be a constructive aid to medical professionals performing virus classification as a way to verify or inform their diagnoses.

9. Reflection

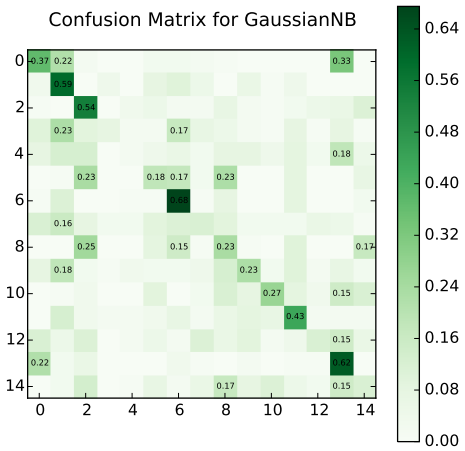
As discussed, due to the high performing nature of our CNN on classifying virus images obtained through microscopy techniques, this particular aspect of virus identification seems worth pursuing. Obtaining this type of data could be difficult, but as we've shown, the number of images obtained for any known virus can be extended with different pre-processing techniques, including rotations, embossing, and other additive noise measures. Clearly this represents a powerful aid for doctors and other members of the medical profession to inform and verify diagnostics. The nature of viruses is that they're constantly evolving, but we'd like to posit that machine learning methods for image classification are not limited to classifying known viral strains. Raj et al. [11] showed sparse, tree-structured models could be learned from decision rules based on genetic subsequences that could predict viral hosts using discriminative machine learning tools. In this instance, the hosts of a novel virus can be determined even if it's remotely similar to a known viral host. It's a worthwhile endeavor to see if a CNN could be trained to predict unknown viruses in the context of training on a set of labeled viral strains from a family, and testing on a particular modified or evolved strain from the same family.

Appendix

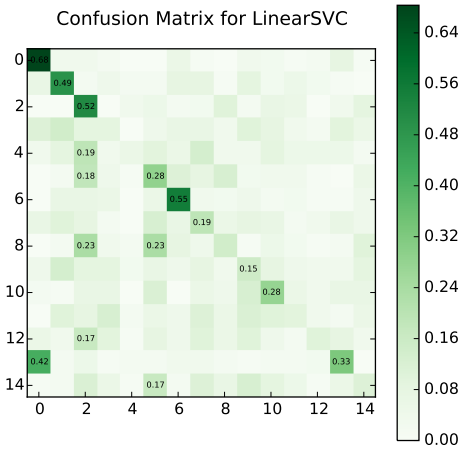
A. Baseline Classification Results



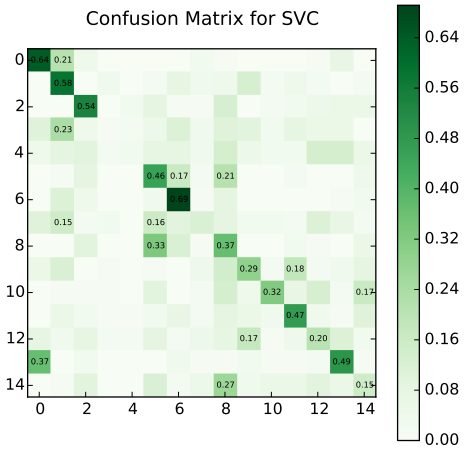
(a) Logistic Regression Confusion Matrix Results



(b) Gaussian Naive Bayes Confusion Matrix Results



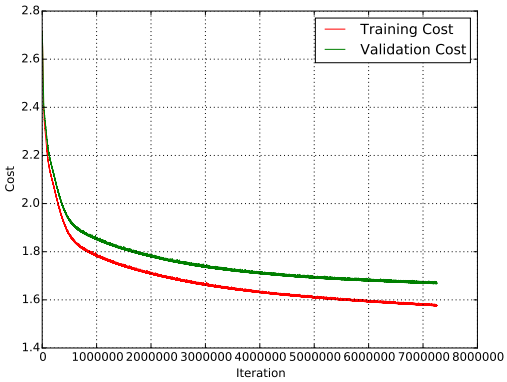
(c) Linear SVM Confusion Matrix Results



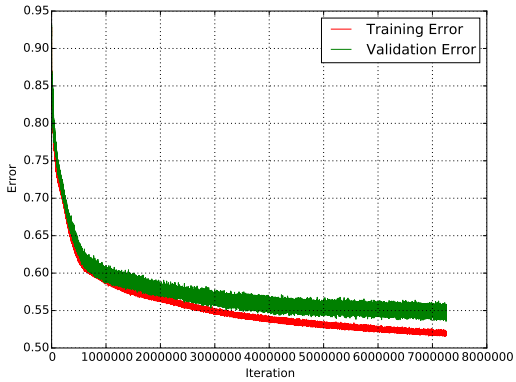
(d) SVM Confusion Matrix Results

Figure 5: Confusion Matrices for the different baseline classifiers.

B. Neural Network Trained on LBP Results

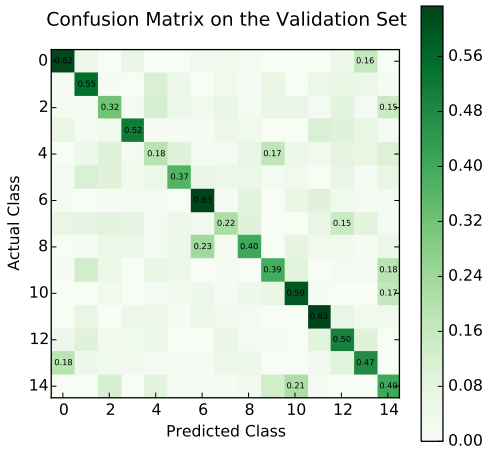


(a) Training versus Validation Cost

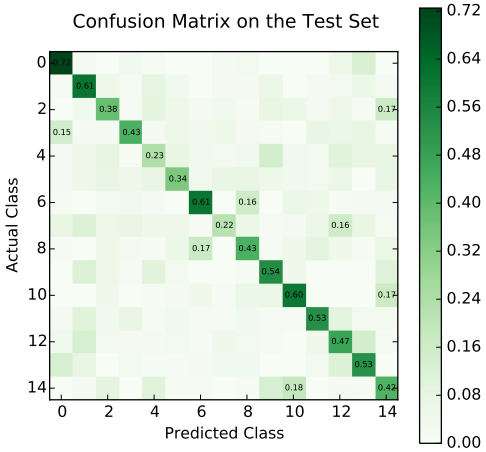


(b) Training versus Validation Error

Figure 6: Learning curves for a feed-forward neural network with one hidden layer of 256 units trained on LBP.

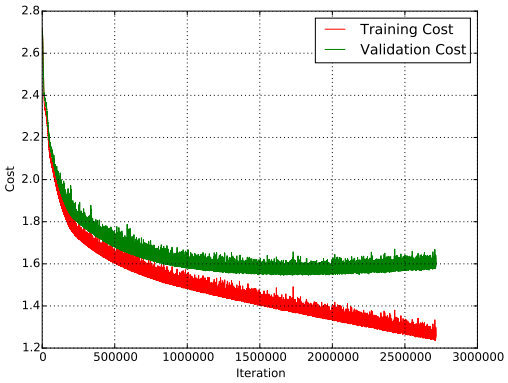


(a) Confusion Matrix on the Validation Set

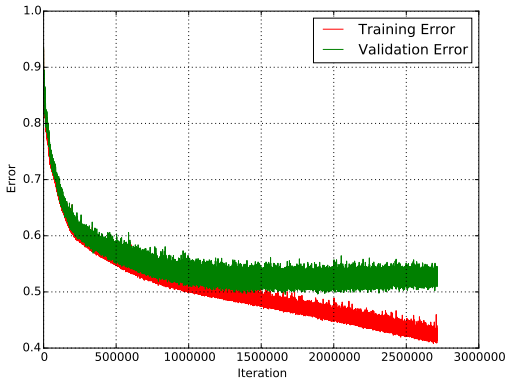


(b) Confusion Matrix on the Test Set

Figure 7: Confusion Matrices for a feed-forward neural network with one hidden layer of 256 unit trained on LBP.

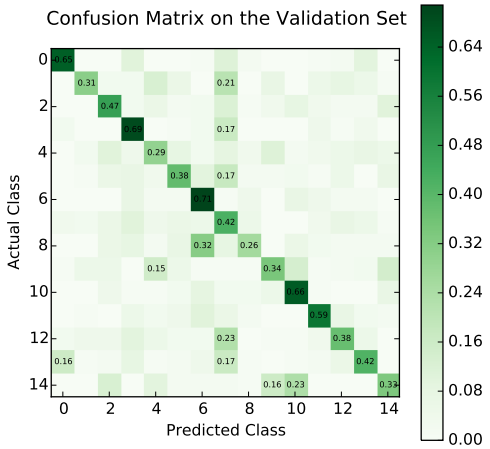


(a) Training versus Validation Cost

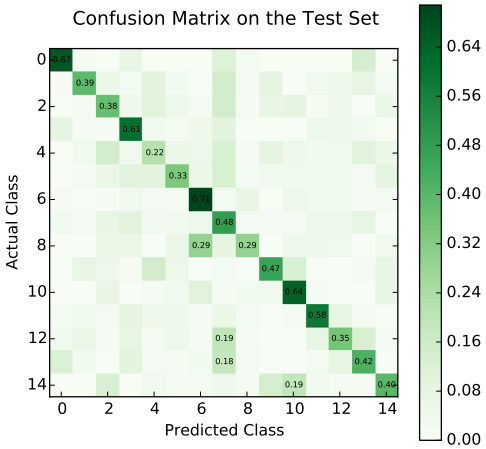


(b) Training versus Validation Error

Figure 8: Learning curves for a feed-forward neural network with two hidden layers of 256 units each trained on LBP.

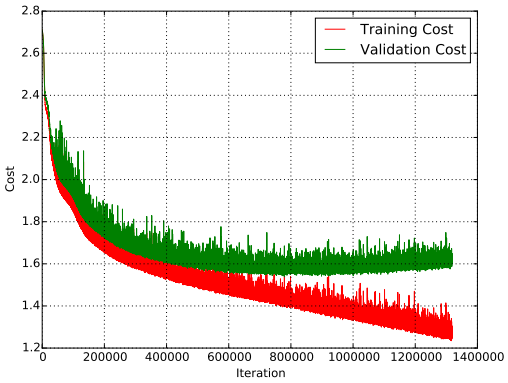


(a) Confusion Matrix on the Validation Set

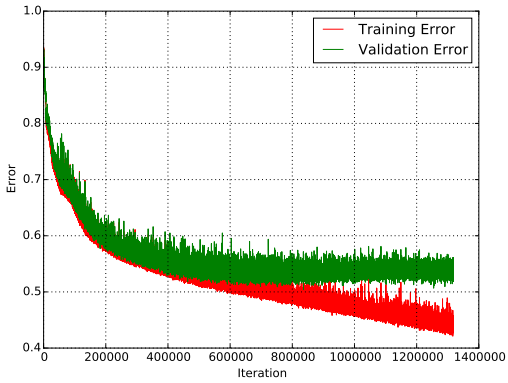


(b) Confusion Matrix on the Test Set

Figure 9: Confusion Matrices for a feed-forward neural network with two hidden layers of 256 units each trained on LBP.

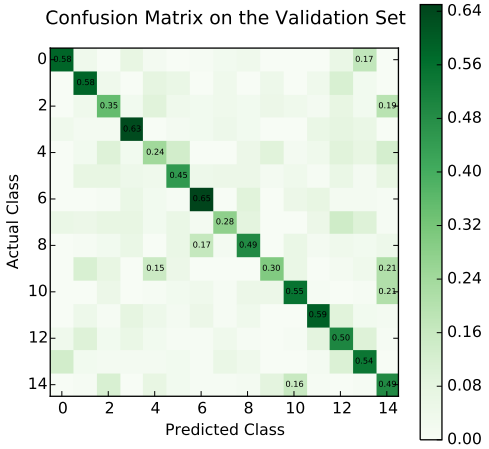


(a) Training versus Validation Cost

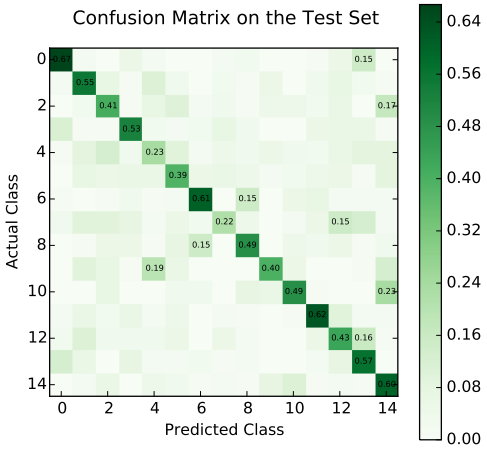


(b) Training versus Validation Error

Figure 10: Learning curves for a feed-forward neural network with three hidden layers of 256, 128, 64 units trained on LBP.



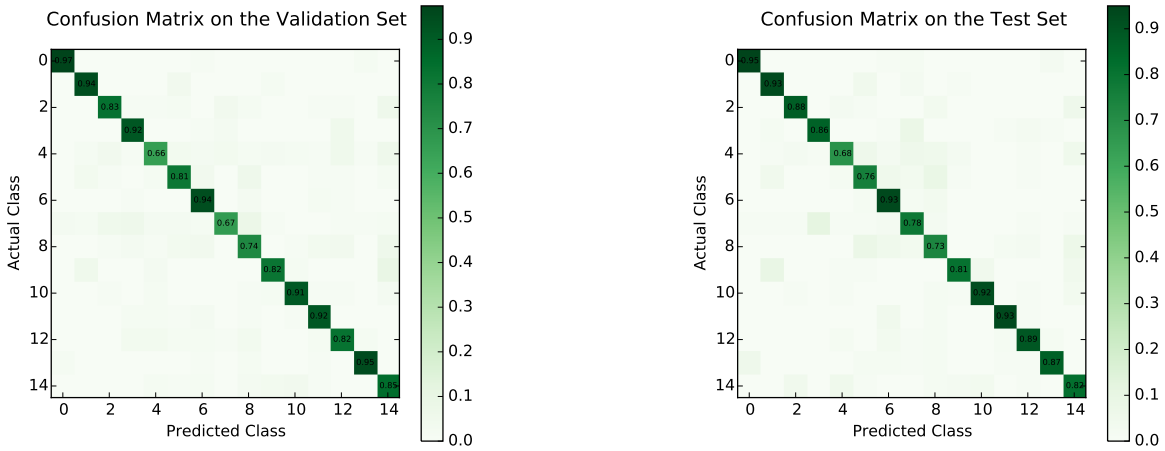
(a) Confusion Matrix on the Validation Set



(b) Confusion Matrix on the Test Set

Figure 11: Confusion Matrices for a feed-forward neural network with three hidden layers of 256, 128, 64 units trained on LBP.

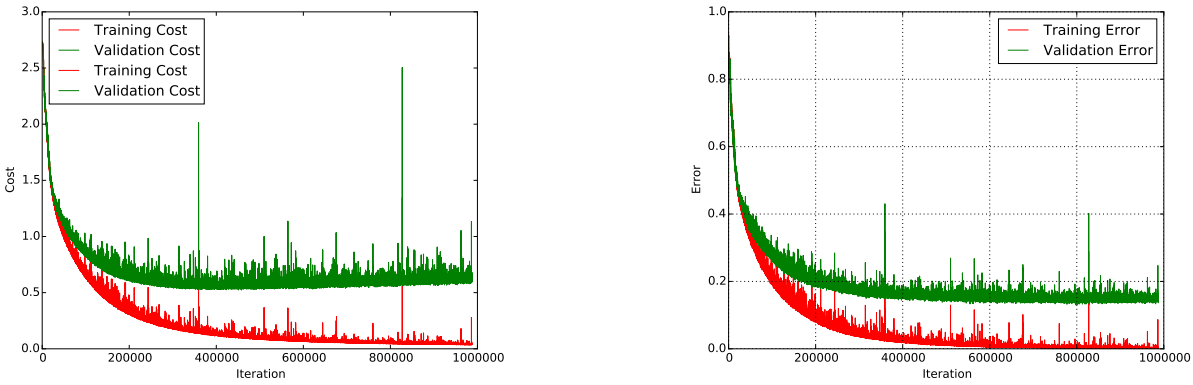
C. Convolutional Neural Network Results



(a) Confusion Matrix on the Validation Set

(b) Confusion Matrix on the Test Set

Figure 12: Confusion Matrices of our Convolutional Neural Network with three convolutional layers



(a) Training versus Validation Cost

(b) Training versus Validation Error

Figure 13: Learning curves for our Convolutional Neural Network with three hidden layers

We hereby state that all the work presented in this report is that of the authors.

Alan wrote the LBP explanations and *Neural Network Approach*. Alan programmed the feed-forward neural network in `emerald.py` and extracted the Local Binary Patterns from the dataset via `purify_dataset.py`.

Kian created the rotated dataset and wrote its section *Transformed Dataset*. He also wrote the *Baseline Classification Approach* and *Random Forest Approach* sections and programmed the baseline learners.

Genevieve wrote the *Introduction*, *Problem Definition*, *Convolutional Neural Network Approach* and *Reflection* sections.

Alan and Genevieve wrote the *Related Work* and *Methodology* sections together.

Genevieve programmed the CNN with help from Alan's `emerald.py`, and Kian helped debug.

References

[1] Pavan K Attaluri, Ximeng Zheng, Zhengxin Chen, and Guoqing Lu. Applying machine learning techniques to classify h1n1 viral strains occurring in 2009 flu pandemic. *BIOT-2009*, 21, 2009.

[2] Pavan K Attaluri, Zhengxin Chen, and Guoqing Lu. Applying neural networks to classify influenza virus antigenic types and hosts. In *Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 2010 IEEE Symposium on*, pages 1–6. IEEE, 2010.

[3] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[4] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[5] Luis Pedro Coelho. Mahotas: Open source software for scriptable computer vision. *arXiv preprint arXiv:1211.4907*, 2012.

[6] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay D. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *CoRR*, abs/1312.6082, 2013. URL `http://arxiv.org/abs/1312.6082`.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[8] G. Kylberg, M. Uppström, K.-O. Hedlund, G. Borgefors, and I.-M. Sintorn. Segmentation of virus particle candidates in transmission electron microscopy images. *Journal of Microscopy*, pages no–no, 2011. ISSN 1365-2818. doi: 10.1111/j.1365-2818.2011.03556.x. URL `http://dx.doi.org/10.1111/j.1365-2818.2011.03556.x`.

[9] Gustaf Kylberg, Mats Uppström, and Ida-Maria Sintorn. Virus texture analysis using local binary patterns and radial density profiles. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 573–580. Springer Berlin Heidelberg, 2011.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] Anil Raj, Michael Dewar, Gustavo Palacios, Raul Rabadan, and Christopher H Wiggins. Identifying hosts of families of viruses: A machine learning approach. 2011.

[12] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61: 85–117, 2015.