Classifiying Transmission Electron Microscopy Virus Textures

Alan Do-Omri   Kian Kenyon-Dean   Genevieve Fried

260532985   260564475   260564432

December 13, 2015

Abstract

On a daily basis doctors are tasked with determining the identity of viruses from blood samples. Even with advanced microscopy techniques, viruses can be difficult to discern by the expert human eye. Unfortunately, Misclassifying can have grave implications, not only for the patients in question but also for those who come into contact with him or her. Fortunately, virus classification is an image classification problem, and machine learning methods present exceedingly high accuracy rates for image classification tasks and don't require 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 [14]. 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. [7], and ImageNet classification Krizhevsky et al. [8].

The high degree of accuracy obtained with these automated methods is highly desirable for a problem like virus classification. In the desire to determine a model that accurately predicts known virus types, we treat the virus classification as an image classification problem. We use the Virus Texture Dataset from Uppsala University containing images of fifteen different virus types, which we extend through random rotations to enlarge our dataset.

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 methods to accurately classify viruses from viral sample images. This extends to both viruses which are cur-

1

rently 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. [13] Attaluri et al. [1] Attaluri et al. [2]. Kylberg et al. [10], which we looked to for insight into feature extraction methods, use 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. Methodology

3.1. Feature Design and Selection Methods

3.1.1 Related Work

In the work of Kylberg et al. [10], texture analysis is performed on a dataset composed of 22 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, such as Local Binary Patterns, Radial Density Profile and their respective variants before proceeding to detail their results. We will briefly explain their feature extractors and results here.

3.2. Local Binary Pattern (LBP)

Local Binary Pattern works such that, 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 1 – 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.
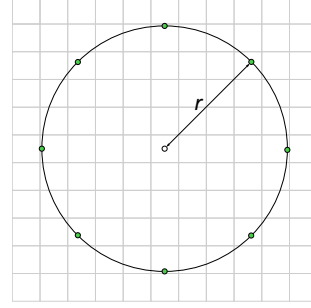

Figure 1: Example of LBP sampling: The green points are the neighbouring sample points at distance r from the central white point. In this case, we are sampling with $n = 8$ neighbour points.

A sequence of 0s and 1s are constructed $v(p_i)$ to form a binary number v_{p_i}, hence the name. 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. [10] 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.

Rotational Invariant LBP In order to reduce the size of the feature vector, Kylberg et al. [10] mention a modification of LBP in the following sense: instead of creating v_{p_i} by interpreting the vector $v(p_i)$ and using v_{p_i} as is, rotate the number v_{p_i} bitwise until you get the smallest possible number. For example, the number 110 becomes 011. They name this technique the rotational invariant and denote it with $\text{LBP}^{\text{ri}}_{n,r}$.

Uniform LBP To further reduce the size of the histogram, Kylberg et. al also restrict the values of v_{p_i} to only numbers that have 2 or less transitions from 0 to 1 or from 1 to 0, and they call this variant *uniform binary patterns with at most 2 spatial transitions*, denoted $\text{LBP}^{\text{u2}}_{n,r}$. For example, 01010 has 2 transitions from 0 to 1 and 2 transitions from 1 to 0 which makes 4 transitions in total. This version then $\text{LBP}^{\text{u2}}_{n,r}$ does not count the number 01010. On the other hand, 00111 has only 1 transition from 0 to 1 so it is accepted.

Gaussian Filtered LBP Finally, Mäenpää and Pietikäinen [11] talk about an extension to the LBP that is also used in the work of Kylberg et al. [10], which consists of sampling neighbours about a central pixel at different radii. Instead of sampling these

points equidistantly in a circle, the authors sample according to a Gaussian distribution. Figure 3 shows an example of such a sampling. Kylberg et al. [10] denote it as $\text{LBPF}^{\text{ri}}_{n_1,r_1} + ^{\text{ri}}_{n_2,r_2} + \cdots + ^{\text{ri}}_{n_j,r_j}$ for the different numbers of sample points $n_1, n_2, \cdots, n_j$ and different radii distances $r_1, r_2, \cdots, r_j$.

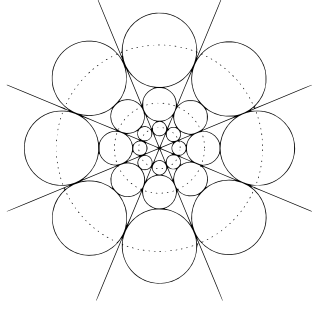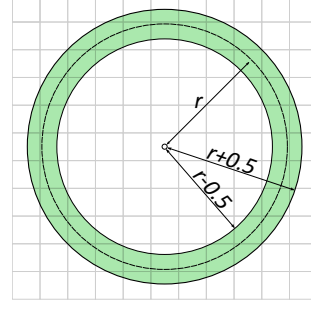Figure 3: Example of RDP sampling: The green zone is the set of sampled pixels for a given radius r.

It is not explained how the pixel features are combined to make the feature vector for the whole image but we would imagine that the feature vector for the whole image is either a concatenation of the pixel features or the concatenation of the mean of each pixel feature.

Fourier RDP One variation Kylberg et al. [10] introduced is applying the same RDP technique on the image after it undergoes a Fourier transform and looking at it with respect to a base e logarithmic scale. They denote it with FRDP_n.

3.4. Results

The first thing to note is that Kylberg et al. [10] made modifications to the images' scales. In one case, one pixel in the image corresponded to 1 nanometer, they refer to that scale as *fixed scale*. In the other case, the radius of the virus is set to take 20 pixels, they refer to that scale as *object scale*. The classifier used was a Random forest with 100 trees (they tried up to 200 with no significant improvement with respect to 100 trees). The different textures Kylberg et al. [10] used were among

1. $\text{LBP}^{\text{ri}}_{8,2}$
2. $\text{LBP}^{\text{riu2}}_{8,2}$
3. $\text{LBPF}^{\text{ri}}_{8,1} + ^{\text{ri}}_{8,2.4} + ^{\text{ri}}_{8,5.4}$
4. $\text{LBPF}^{\text{riu2}}_{8,1} + ^{\text{riu2}}_{8,2.4} + ^{\text{riu2}}_{8,5.4}$
5. RDP_{20}
6. FRDP_{20}

All values are chosen after a grid search. Figure 4 shows their results with respect to the six aforementioned texture extractors and on both fixed and object scale images. They conclude that $\text{LBPF}^{\text{ri}}_{8,1} + ^{\text{ri}}_{8,2.4} + ^{\text{ri}}_{8,5.4}$ performed the best in fixed scale (median 21% error) whereas FRDP_{20} performed the best in object scale (median 22% error). They also noticed that combining these two texture extractors, they could get an even

Figure 2: Example of LBPF sampling: In the simple LBP, the points would be sampled at equal distances around the dotted circles for one radius as in figure 1 but with this extension, the points are sampled according to a Gaussian distribution inside the solid black circles at multiple radii. In this picture, the number of neighbours also vary with the radii. This image comes from Mäenpää and Pietikäinen [11].

3.3. Radial Density Profile (RDP)

This method is a way to get the mean intensity in a ring around each pixel in the image. Kylberg et al. [10] defines the *radial mean intensity* f around a center pixel q_c as

$$f(q_c, r) = \frac{1}{|N|} \sum_{q \in N} I(q)$$

where $I(q)$ is the pixel value for pixel q and $N = \left\{ q : \|q - q_c\|_2 \in \left(r - \frac{1}{2}, r + \frac{1}{2} \right] \right\}$ is the set of pixels in a ring around q_c of width 1 at distance r from q_c. Figure 3 shows an example of what the set N may look like. Following the notation from Kylberg et al. [10], let $\bar{f}_{q_c}$ be the mean of the set $\{f(q_c, i)\}_{i=1,\cdots,n}$. Then they define the radial density profile for that pixel q_c as $\text{RDP}_n = \left[f(q_c, 1) - \bar{f}_{q_c}, f(q_c, 2) - \bar{f}_{q_c}, \cdots, f(q_c, n) - \bar{f}_{q_c} \right]$.
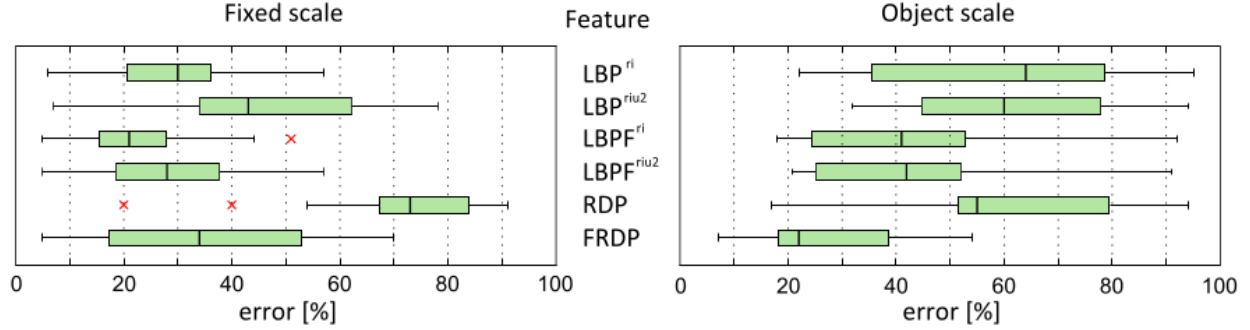
Figure 4: This figure comes from Kylberg et al. [10]. The classification errors are shown for a Random Forest classifier using the 6 different texture extractors as described in section 3.4. The boxes' vertical lines represent the median and the red × represent outliers that are at least 1.5 times the size of the box away from it. The error bars are from the lower to the upper quartile.

better result (median 13% error).

3.5. Data 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. [9] 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. It is a rather small dataset for the purposes of our CNN so we chose to extend it by generating twelve random rotations for each image in the dataset, producing 18,000 more images.

4. Baseline Classification Approach

We tested 4 different classifiers using algorithms from the scikit-learn Pedregosa et al. [12] machine learning library. These baseline classifiers provide results that help to reveal the dataset's complexity. We trained and tested these classifiers over the rotated dataset. Not surprisingly, the SVM with an rbf kernel performed the best since it allows projection of the pixel feature space to a higher dimension, thus making it capable of capturing aspects of the data that cannot be observed by the other classifiers. These classifiers clearly have poor results, but they provide a good baseline understanding of the dataset, showing that

the image pixel space is not easily separable and requires a stronger non-linear function approximator in order to be properly understood. See the appendix for the confusion matrices of the results.

5. Neural Networks Approach

In order to establish a baseline, we combine ideas from Kylberg et al. [10] and ourselves by attempting to use a neural network with the LBP features. Our neural networks are built using the Lasagne and Theano [3, 4] libraries. In our experiments, we will not rescale the images and we will use $LBP_{8,2}$, an implementation given by the *Mahotas* computer vision library Coelho [6]. We are not scaling the images because we are later comparing to CNNs, which will not use rescaled images. The values of $8, 2$ come from the best values as checked by Kylberg et al. [10]. *Mahotas*' implementation of LBP is done so 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. This is achieved by some optimization on their end. 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 more certain that no features in the normalized test set would have a value greater than 1. In all cases, we are using a learning rate of 0.005 and an L2 regularization weight of 0.0001, both arbitrarily chosen and the last layer was a 15 units softmax layer. The dataset described in section 3.5 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 found in figures 6 and 7; the second neural network results are found 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.

5.1. Results

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

5.2. Discussion

We had to stop the training of the one hidden layer neural network due to time constraints. We notice that the neural network with one hidden layer takes the longest time to converge whereas the one with three hidden layers took the shortest time. On the other hand, the network with three hidden layers overfitted the quickest. This is to be expected since it has many more parameters than the network with one layer. We also notice that the network with one hidden layer has the biggest error and we think this is because the network does not have enough neurons to learn properly. The three hidden layers network has too many neurons and is overfitting but we think its results can be improved by adding Gaussian noise to neurons and dropping out some of them. Nonetheless, it is still the network that gave the best results.

Compared to the work of Kylberg et al. [10], we can see from Figure 4 that the LBP$^{\text{ri}}$ 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. Finally, *Mahotas'* implementation of LBP does not necessarily match the implementation of LBP$^{\text{ri}}$ so this can induce some error as well.

6. 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 approach, CNNs seemed a natural method to use. Our CNN is build with the Lasagne and Theano [3, 4] libraries. Our CNNs 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 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.

6.1. Results

Images pertaining to the results of our CNN 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 %

6.2. Discussion

Our CNN classifies viruses on our training set with an accuracy of 85%. Due to the critical nature of the task, this is not as high as we had hoped for a standalone virus classifier, but certainly high enough to be a constructive aid to medical professionals performing virus classification (to verify or inform their diagnoses). 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 nuber of convolutional layers, percentage of dropout, learning rate, to name a few, is a worthwhile investment given more time and effort.

In addition, in the last few years several papers have shown that ensemble methods perform very well for reducing testing error Chen [5]. While it is not understood why ensembles of CNNs perform so well, using these methods in our classification task might have improved performance.

**** FINISH LATER

7. Reflection

As discussed, due to the high performing nature of our CNN on classifying a virus from an image of it 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 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 just limited to classifying known virus strains. Raj et al. [13] showed a sparse, tree-structured models could be learned from decision rules based on genetic subsequences for predicting viral hosts using discriminative machine learning tools. In this instance, the hosts of a novel virus can be determined even if it shares distant similarity with 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 labelled viral strains from a family, and testing on a particular modified or evolved strain from the same family.

Appendix

A.   Baseline Classification Results

B.   Neural Network Trained on LBP Results

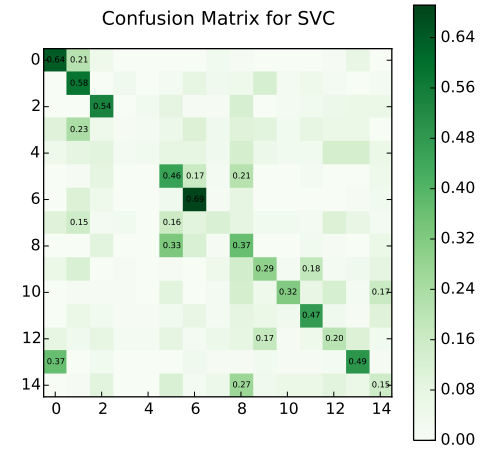## C.   Convolutional Neural Network Results
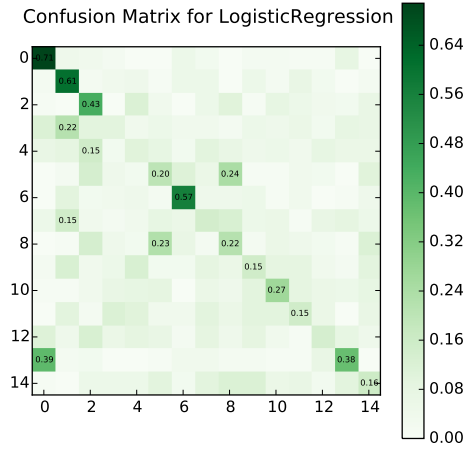
(a) Logistic Regression Confusion Matrix Results
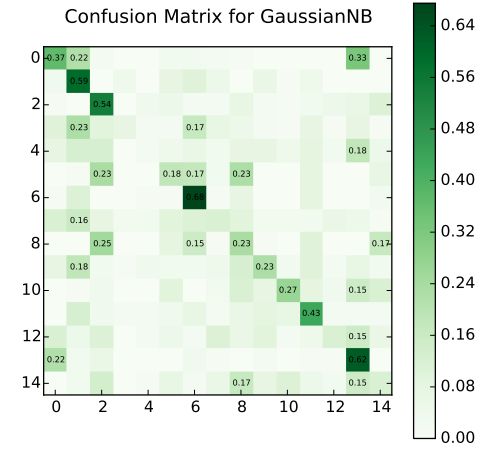
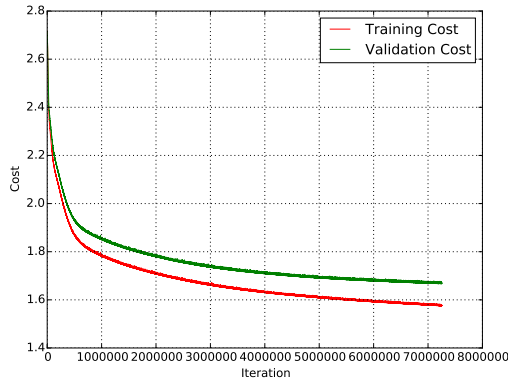(b) Gaussian Naive Bayes Confusion Matrix Results
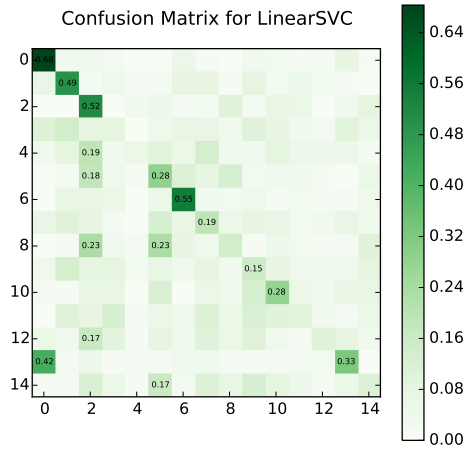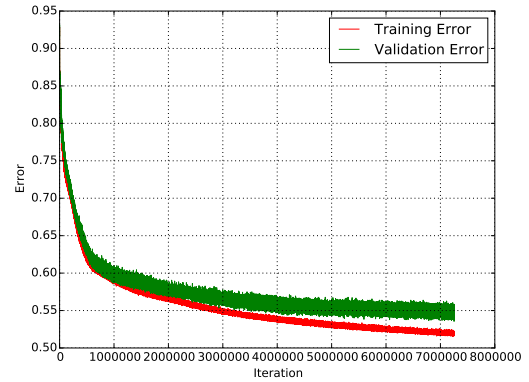

(c) Linear SVM Confusion Matrix Results

(d) SVM Confusion Matrix Results

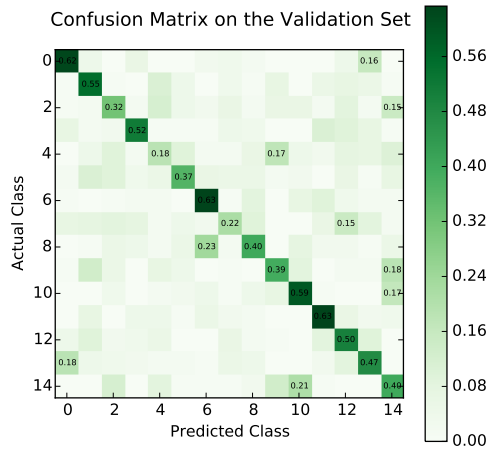Figure 5: Confusion Matrices for the different baseline classifiers.
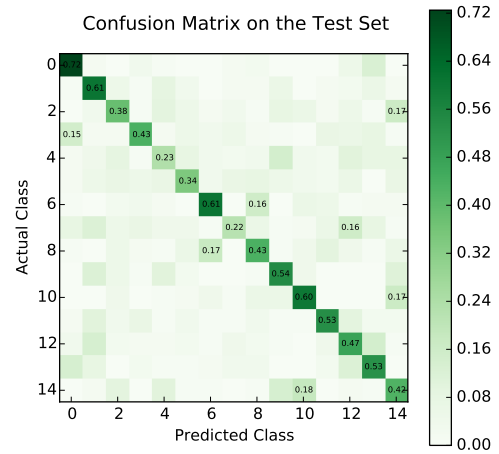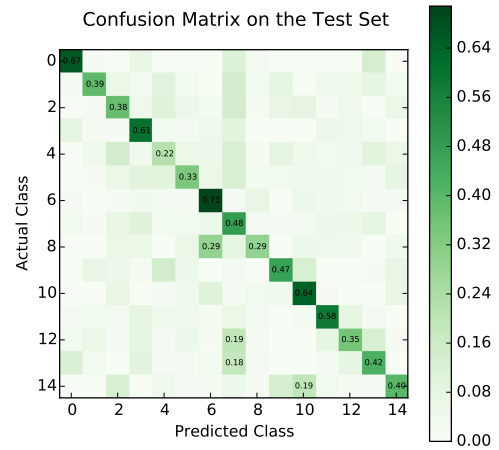
(a) Training versus Validation Cost

(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.

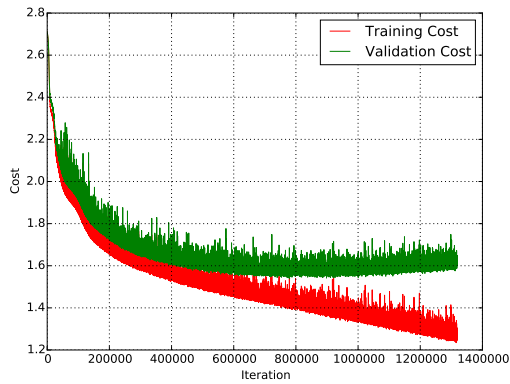

(a) Confusion Matrix on the Validation Set

(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.
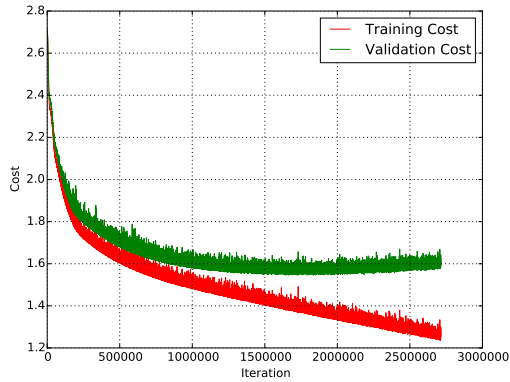
(a) Training versus Validation Cost
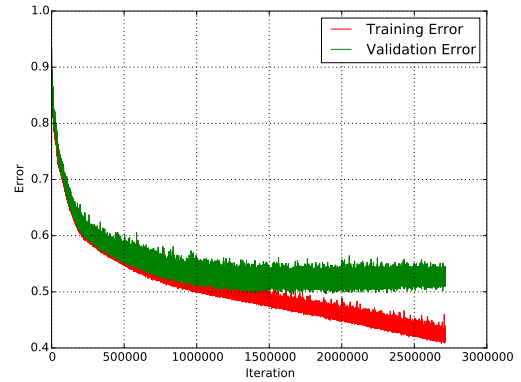
(b) Training versus Validation Error

Figure 8: Learning curves for a feed-forward neural network with two hidden layers of 256 units each trained on LBP.
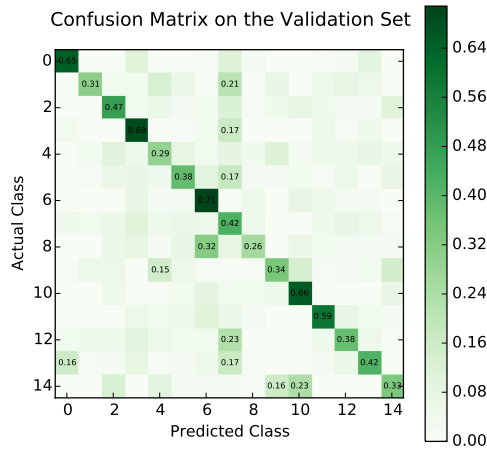
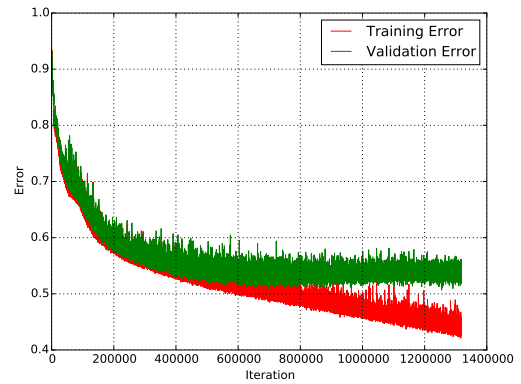(a) Confusion Matrix on the Validation Set

(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.
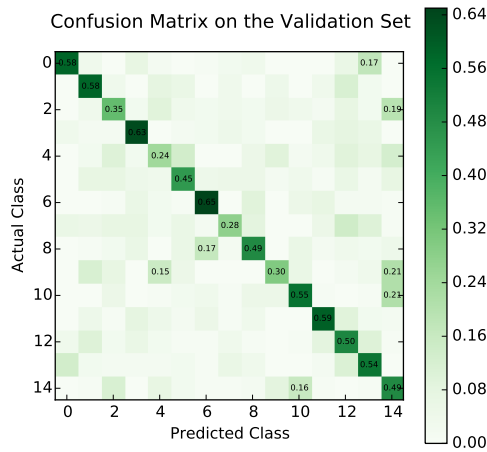
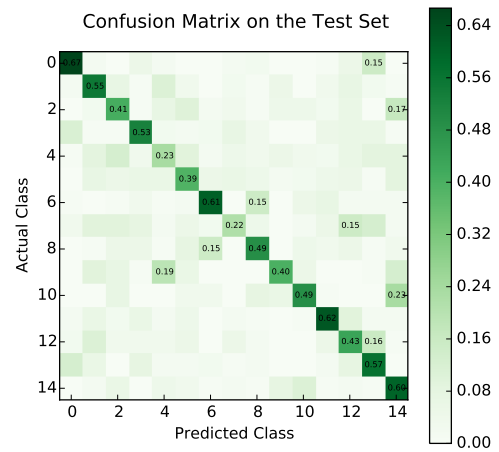(a) Training versus Validation Cost

(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.
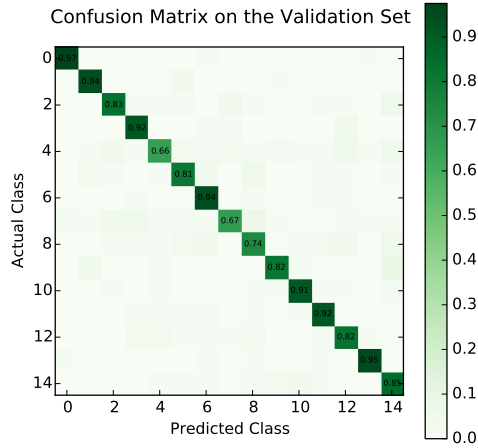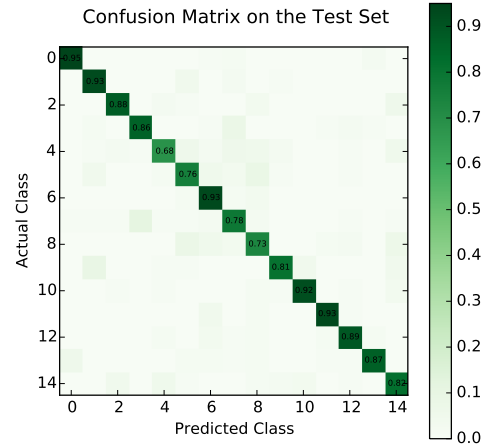

(a) Confusion Matrix on the Validation Set

(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.
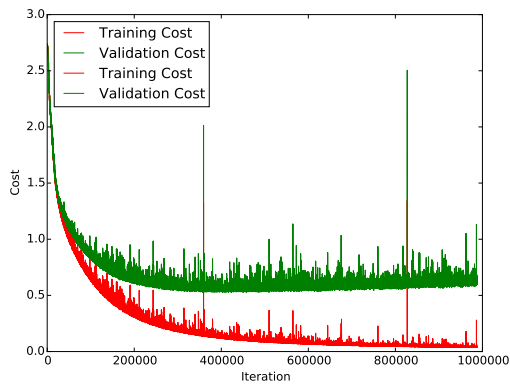
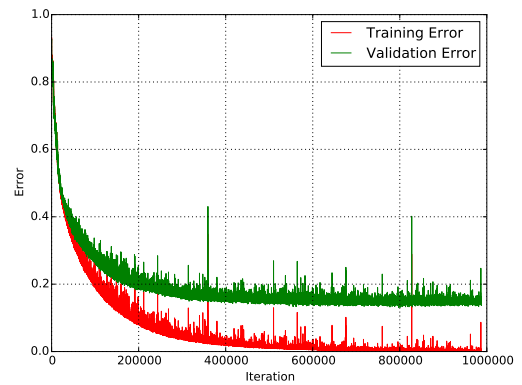(a) Confusion Matrix on the Validation Set

(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

12

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

Alan wrote the sections for *Related Work* and *Neural Network Approach*. Alan programmed the feed-forward neural network in `emerald.py` and extracted the Local Binary Patterns from the dataset in `purify_dataset.py`.

Kian created the rotated dataset and wrote its section *Transformed Dataset*, programmed the baseline learners and wrote the content for *Baseline Results* and *Conclusion*.

Genevieve wrote the *Introduction*, *Problem Definition* sections, and the results and discussion of CNN.

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

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] Liran Chen. Learning ensembles of convolutional neural networks.

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

[7] 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`.

[8] 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-`pdf.

[9] 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`.

[10] 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.

[11] Topi Mäenpää and Matti Pietikäinen. Multi-scale binary patterns for texture analysis. In *Image Analysis*, pages 885–892. Springer, 2003.

[12] 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.

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

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