Sum of Two Handwritten Digits: A Machine Learning Classification Problem

Team: Do Neural Networks Dream of Strictly Convex Sheep?
Pulkit Khandelwal pulkit.khandelwal@mail.mcgill.ca, 260717316
Elisabeth Sulmont, elisabeth.sulmont@mail.mcgill.ca, 260567973
Carlos G. Oliver, carlos.gonzalezoliver@mail.mcgill.ca, 260425853

***Abstract*— The aim of this project is to classify images containing two hand-written numbers into categories according their sum. We have have trained a large set of learning algorithms and parameters to solve this task and hereby present the obtained results. The study is performed as a component of Project 3 of COMP551: Applied Machine Learning class at McGill University.**

I. INTRODUCTION

Handwritten digit recognition (classification) has been an active area of research in the computer vision community from as early as the 1990s with Yann LeCun's neural networks [1]to the latest state of the art algorithms including deep networks [3]. We present the results of our study on an image classification task where given a grayscale image (which contains two digits) we are to predict the sum of the given two digits. Descriptors such Histogram of Oriented Gradients and Daisy are used for feature representation for some baseline classification algorithms such as Logistic Regression and Support Vector Machines. We then propose some more sophisticated algorithms such as: sliding window approach on MNIST data, feed-forward neural networks, Convolutional Neural Networks as well as Recurrent Neural Networks to experiment, validate and finally make predictions on the given dataset of 100,000 images of 60X60 pixels.

II. RELATED WORK

Instead of using cross-validation and trial and error for each method, we looked at some of the benchmark papers and thereby use the knowledge from the existing work in the field to our advantage. Here we review some of the literature on the subject.

A. Feature Engineering

Many of the traditional vision problems have focused on feature representation. David Lowe [4] exploits the scale space theory through the repeated use of Gaussian kernels to give rotation and scale invariant SIFT descriptors. Image matching and stitching problems have readily been using these descriptors. Speeded up Robust Features [5] proposed by Tinne Tuytelaars et. al. claims to produce more robust, repeatable and distinctive interest points than previously known detectors. Dalal and Triggs[6] proposed a state of the art SVM learning model for human detection and classification through the use of HOG descriptors. In this work, we used Daisy features which are a modified version of SIFT descriptors and also compared the results with HOG descriptors. Not all descriptors convey essential information so it is a good practice to use dimensionality reduction using Chi-square. We use Chi-square test because it performed better than PCA and Information Gain in the previous Project.

B. Baseline algorithms: Logistic Regression and Support Vector Machine

A multiclass regression problem can be thought of as a maximum entropy formulation using a softmax function and gives an example of digit recognition using Logistic Regression on the original MNIST dataset which achieves an accuracy of 93.62%. Dennis Decoste et al.[9] incorporates prior knowledge about invariances of a classification problem and reports the lowest error rate of 0.6% on the MNIST dataset at the time using a SVM model. We adapt a variant of this model for the SVM classification in our study.

C. Feed-forward neural network and Deep Learning: CNNs and RNNs

[15] gives a list of results of different approaches to the original MNIST classification with the lowest error obtained of 0.21% as obtained by Rob Fergus et al . They introduce DropConnect which regularizes large fully connected layers within neural networks. LeNet by LeCunn et al.[1], AlexNet by Alex Krizhevsky et al.[17] , VGGNet by Karen Simonyan et al.[3] and GoogLeNet by Christian Szegedy et al.[8] have been successful at showing incremental progress over the years using deep ConvNets on various image recognition and classification problems. Quoc V. Le et al[18]explains a simple way to initialize Recurrent Networks of Rectified Linear Units. We extend this for the digit classification problem as explained in[14]. We

draw inspiration from these papers and propose our model.

III. PROBLEM REPRESENTATION

A. Preprocessing

For the baseline algorithms and feed-forward neural network, the images were normalized between zero and one and also left un-normalized. We present the results for the un-normalized version to its stronger performance on the validation set.

B. Feature selection and extraction

After a thorough literature review we chose to extract HOG and Daisy descriptors due to their reliability over others, and because they were much more robust to scale and rotation invariance. Scikit-image[12], vlfeat[10] and Python Imaging Library[13]were used for the same. For daisy descriptors each image resulted in 1x104 dimension vector as each feature was of 8 orientations, radius of 20 units and 2 such features were detected for each image. For HOG descriptor each image resulted in 1x3600 dimension vector with a cell size of 4x4 and 16 orientations. We also flattened out the original image of 60x60 to 1x3600 and presented the results for the same. 3 shows one image from the given dataset which has two digits. 4 shows the Daisy features and its descriptors with 6 histograms. 5 shows the HOG descriptors for a cell size of 4x4 pixels and 16 orientations. 6 shows the SIFT features and their descriptors.

C. Feature Reduction

Chi-square test was used to reduce the dimension for the HOG descriptors and retain the top 500 features.

*Note:*For CNNs and RNNs the images are not preprocessed in any way and taken in the raw format where each image is 60x60.

IV. ALGORITHM SELECTION AND IMPLEMENTATION

We tested many variants of each of the seven techniques/algorithms. Here we give briefly give the precise details. Given that many of these approaches are computationally expensive, we did not apply k-fold validation to select the best model. Instead we applied a randomly shuffled training set of 80% and validation set of 20%.

A. Sliding Window Technique

We train a standard feed-forward neural network from `sklearn` and a simple convolutional neural network from `keras` on the MNIST dataset to recognize individual digits in our images through a window scanning

B. Logistic Regression

We used sklearn's Logistic Regression with regularization hyper-parameter λ. *L2* regularization was used.

C. Support Vector Machine

A multiclass SVM was trained using a Radial Basis Function kernel.

D. Feed-forward Neural Network

We implemented a fully connected feedforward neural network trained by backpropagation. In an attempt to have a computational advantage, a stochastic gradient descent was specifically implemented using mini-batches, as recommended and explained in *Neural Networks and Deep Learning* [19]. This means that in one epoch, we split our training data into mini-batches (random small subsets with 500-1000 examples) and iterate through them. In each iteration, we estimate and update the gradient over the average error of a mini-batch. The intended effect being a faster convergence than doing batch gradient descent or stochastic gradient descent over single examples.

E. Convolutional Neural Network

We use the Keras [14] python library on top of tensorflow[2] to implement several convolutional neural networks. We choose this method as it has been shown to be successful in learning patterns in a spatially independent manner with minimal preprocessing. The networks are built with varying numbers of convolution, and pooling layers and finally sent through a number of fully connected neural network layers to produce predictions. We have zero-padded the images. The activation used is *relu* as it is the most widely used activation function in the literature. The final classifier is built using a *softmax* classifier, the optimizer used is *adam*. These choice are obvious due to their popularity in terms of their performance. More discussion on the network architecture follows in Section V of this report.

F. Recurrent Neural Network

To further explore the field of Deep Learning we came across an implementation of IRNN experiment on pixel-wise-pixel sequential MNIST as mentioned in [18]. We reproduce the experiment as given in[18].

V. RESULTS: TESTING AND VALIDATION

A. Sliding Window Technique

Our first intuition was to break down to problem into problems that we know how to solve efficiently. So we decided to use the MNIST dataset, which is the source dataset for the 'difficult digits set' that has been extensively studied and for which there exist very effective models. We therefore took

a 'sliding window' approach to break down the images into 28x28 pixel images with a step size of 5 pixels that we could send to a neural network trained on the MNIST dataset. The MNIST neural net would then return a set of probabilities for each possible digit 0-9. We retain the digit with the highest probability and once all windows are scanned, we take the two windows with the highest probability guess as the two digits in the original image, and the algorithm finally returns their sum. In the case that only a single digit is guessed for the entire image, we assume that there were two of the same digits in the original image, so we just return the digit multiplied by 2. In order to avoid windows that are likely to contain background, we omit windows with an average intensity that is lower than the average intensity of the whole image. We first trained a simple neural network (NN) on the MNIST dataset: a simple feed-forward multi-layer perceptron with a single hidden layer of size 50 on which achieved 98.8% test and 98.7% training accuracy after 10 epochs. When we fed the windows from the two digit images, however the method was only able to accurately classify 7.66% of the training set of full images. It is therefore likely that the NN was overfitting to the MNIST data without the background noise, and with the digits being well centered in the image. With that in mind, we trained a convolutional neural network (CNN) hoping that it would reduce the spatial bias observed in the simple NN. We again trained on the MNIST dataset, this time with two convolution- 'relu' activation steps, followed by a max pooling and dropout step. The resulting images were then fed to a fully connected dense NN layer with a softmax activation. We trained for 12 epochs and achieved 99.25% test accuracy. The CNN performed significantly better on the original two digit images achieving an accuracy of 21.53%. While this is an improvement, it was still not enough to overcome the difference between the training and test images so we did not pursue this method further. However, we did learn that CNNs would be a powerful tool for a task where spatial arrangement is an important factor.

B. Logistic Regression

We didn't explore any hyper parameters for this model as that has been done in the previous reports. We use scikit's implementation. The data was shuffled and the train-validation split was taken as 80:20. Table IV reflects upon the inefficiency of logistic regression. HOG descriptors outperforms Daisy features which in turn performs better than Raw pixels. An accuracy of around 10% calls for a better model.

C. Support Vector Machine

Seeing the inefficiency of Logistic Regression we implemented a multiclass SVM classifier. SVM classifier dominated the classification algorithms before the advent of Deep Learning. the results as reflected in IV shows that SVM is slightly better than Logistic Regression but nowhere close to put this algorithm for practical use for this problem. We then try our hands on Feed-forward Neural Network.

D. Feed-forward Neural Network

To narrow down the best network architecture and learning rate, we first ran 9 different configurations of a wide range, using the HOG features. We tested three different learning rates: 0.1, 0.5, and 1.0 For network architecture: 2 layers of 5 neurons, 5 layers of 10 neurons, and 3 layers of 100, 50, and 25 neurons, respectively. Mini-batch size and number of epochs was set at 1000 and 10, respectively. Table I shows the accuracy for these configurations.

TABLE I
FIRST COMPARISON

	learning rate		
layers	.1	.5	1.0
5,5	6.16%	9.27%	9.89%
10,10,10,10,10	3.13%	9.94%	9.88%
100,50,25	8.10%	8.99%	9.31%

From the results above, we found that 0.1 was too low of a learning rate. We took the top three of the test above and ran them again, but with 25 epochs and a mini-batch size of 500. This was with the idea of improving accuracy by increasing the frequency that the gradient was updated. Table II below shows our results:

TABLE II
SECOND COMPARISON

	learning rate		
layers	.5	1.0	
5,5	9.74%		
10,10,10	9.88%	9.95%	

Increasing epoch size and reducing batch size did not improve our results significantly. To verify that our algorithm was correctly implemented, we can compare a Neural Net implemented by scikit-learn using HOG feautres. On Table III, we see that the results ranged between 9.53%-10.19%. Note that Raw pixels and Daisy features also performed similarly. We can conclude that a Feedforward Neural Network does not perform well in solving this problem, compared to the other methods

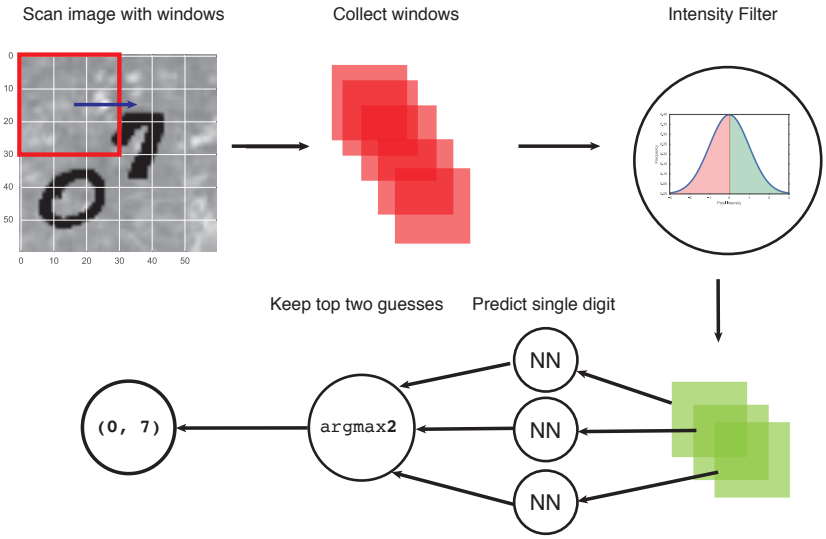

Fig. 1. Illustration of the window scanning process. We decompose each image into overlapping windows of size 28x28. We then remove windows with a lower mean intensity than the mean intensity of the original image in order to reduce the number of noisy windows. We next feed each window to a neural network trained to recognize MNIST single digit images. We retain the top two guesses with the highest probability and output their sum as the prediction.

we tried that have reached >90% accuracy. Also, 12.06% accuracy was obtained for a simple feed-forward neural network implemented in Keras.

TABLE III
SCIKIT-LEARN IMPLEMENTATION OF NN

Hidden Layers	Neurons	Raw pixels	Daisy	HOG
10	10	9.87%	9.43%	10.19%
40	40	10.13%	9.86%	9.53%

E. Convolutional Neural Network

We implemented three models for Deep Neural Network Architectures using ConvNets. A simple model for a head start was implemented as shown in 9 with 2 convolution layers, a maxpool and a dropout layer each and two dense layers. This model was trained on 80:20 data split. We obtained an accuracy of 67.31% after 10 epochs. This convinced us that ConvNets should be explored further. We then implemented a more dense network as seen in 10. We achieved an accuracy of 95.32% on the training dataset and an accuracy of 95.03%. We trained the model for 10 epochs with a batch size of 200. However, we found a caveat. Due to the many layers and convolutions, it can be difficult to understand what exactly is going on in a deep net. In an attempt to visualize the intermediate steps in the network, 7 and 8 show some activation(learned weights) filters inside the network. 7 shows the filters for the original MNIST data and 8 shows the filters for the dataset used in this project. 11 is an example of a more complex ConvNet architecture.

F. Recurrent Neural Network

The original implementation of RNNs claims to achieve 93% train-test accuracy on the original MNIST dataset after 900 epochs [18]. We started implementing this model for our dataset but terminated the program after 18th epoch. The accuracy after 18th epoch was 11.07%. To achieve an accuracy above 90% we should train the network for at least 900 epochs. But, due to obvious time considerations this was not feasible. 12 shows the IRNN architecture.

VI. DISCUSSION

After implementing a large range of learning techniques on the challenging task of double digit recognition, we were able to obtain a correspondingly large range of performances VI.

Our initial attempts using baseline learners such as logistic regression and SVMs performed better than the random baseline however, despite trying various feature selection methods accuracy was still too low to explore these methods further. The exceedingly poor performance is surprising since both these methods have been applied to the MNIST dataset. However, it is likely that the noise and spatial variation of the digits were too complex to fit with these methods. We therefore recognized that there is a strong dependence on the spatial arrangement in the predictions. Wit this in mind, we attempted to split the images through windowing and recognize individual digits using the MNIST source dataset. While a CNN trained on MNIST data was useful in improving accuracy of the windowing method, we see that the CNN is overfitting to MNIST and the noise added to the double

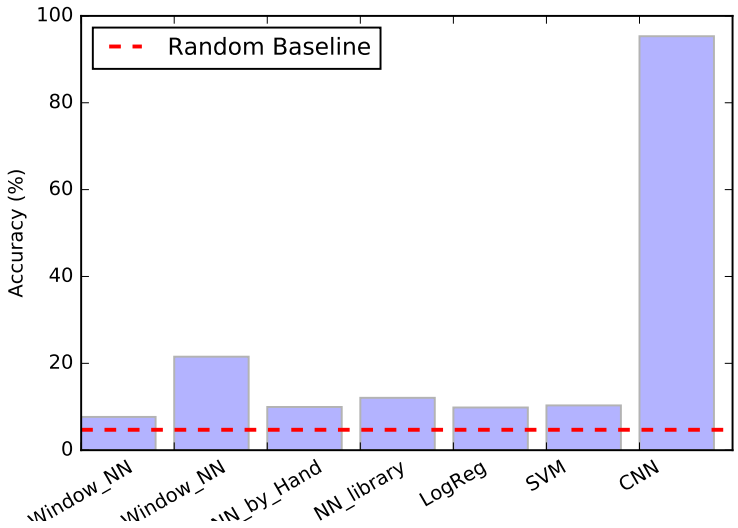

Fig. 2. Summary of accuracy for different trained algorithms on validation set. We draw the expected accuracy of a random classifier as a red dotted horizontal line.

digit dataset was enough to produce weak results. Outperforming both SVM, and Logistic Regression were neural networks who have also been shown to be successful on single digit, low nosie centered images. Yet, similarly to the previously mentioned techniques, they fall short of capturing the patterns in the images. Our highest performing method by far was the CNN. Because feature extraction and spatial dependence are implicitly dealt with in a CNN we saw an explosion in performance, particularly deeper network (with more connected layers) shows stronger performance. We were not able to experiment further with this with many more architectures as CNN training was very computationally expensive. However, we believe that our results will generalize well to the rest of the test set seeing as we had a large dataset and high validation and testing accuracy. To this end, the noise included in the images is likely helping the network generalize better to unseen sets but this is an interesting question to be tested in future work. One weakness of the CNN approach is the difficulty in understanding how the algorithm learns, we just assume it is doing the correct thing when validating. This makes it difficult to know how to improve the networks beyond trial and error, and does not provide much insight into the structure of the data being learned. Current research in the field of text classification is addressing this aspect of neural networks to help provide justification for the learned predictions [20] where the network is able to produce 'rationales' for its predictions. It will be interesting to follow this research for image classification in the coming years.

VII. CONCLUSION

We tested several learning algorithms on the tasks of 'double digit' classification. We show that Convolutional Neural Networks greatly outperform all other methods and achieve accuracy of 95% on the test set. For future work, given enough computational resources, we would explore different network architectures and attempt to better visualize the internal workings of the network.

VIII. STATEMENT OF CONTRIBUTIONS

- **Defining the problem:** Done as a team.
- **Developing the Methodology:** Done as a team.
- **Performing the Data Analysis:** Done as a team
- **Coding the Solution:**
  Logistic Regression,SVM,CNN and RNN- implemented by Pulkit
  Feed-forward Neural Network- implemented by Elisabeth
  Sliding Window Detection - implemented by Carlos

- **Writing the Report:**
  Logistic Regression,SVM,CNN and RNN - Pulkit
  Feed-forward Neural Network - Elisabeth
  Sliding Window Detection, Proofreading - Carlos

  We hereby state that all the work presented in this report is that of the authors unless otherwise referenced

REFERENCES

[1] LeCun, Yann. "LeNet-5, convolutional neural networks." URL: http://yann. lecun. com/exdb/lenet (2015).

[2] Abadi, Martın, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016).

[3] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[4] Lowe, David G. "Distinctive image features from scale-invariant keypoints." International journal of computer vision 60.2 (2004): 91-110.

[5] Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." European conference on computer vision. Springer Berlin Heidelberg, 2006.

[6] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). Vol. 1. IEEE, 2005.

[7] Tola, Engin, Vincent Lepetit, and Pascal Fua. "Daisy: An efficient dense descriptor applied to wide-baseline stereo." IEEE transactions on pattern analysis and machine intelligence 32.5 (2010): 815-830.

[8] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[9] Decoste, Dennis, and Bernhard Schölkopf. "Training invariant support vector machines." Machine learning 46.1-3 (2002): 161-190.

[10] Vedaldi, Andrea, and Brian Fulkerson. "VLFeat: An open and portable library of computer vision algorithms." Proceedings of the 18th ACM international conference on Multimedia. ACM, 2010.

[11] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." Journal of Machine Learning Research 12.Oct (2011): 2825-2830.

[12] Van der Walt, Stefan, et al. "scikit-image: image processing in Python." PeerJ 2 (2014): e453.

[13] Sanner, Michel F. "Python: a programming language for software integration and development." J Mol Graph Model 17.1 (1999): 57-61.

[14] https://github.com/fchollet/keras/blob/master/examples/

[15] https://rodrigob.github.io/

[16] http://neuralnetworksanddeeplearning.com/

[17] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[18] Le, Quoc V., Navdeep Jaitly, and Geoffrey E. Hinton. "A simple way to initialize recurrent networks of rectified linear units." arXiv preprint arXiv:1504.00941 (2015).

[19] Nielsen, Micheal. Neural Networks And Deep Learning. 1st ed. 2016. Web. 7 Nov. 2016.

[20] Lei, Tao, Regina Barzilay, and Tommi Jaakkola. "Rationalizing Neural Predictions." arXiv preprint arXiv:1606.04155 (2016).

APPENDIX

TABLE IV

COMPARISON: BASELINE ALGORITHMS

Model	Raw Pixels	daisy	HOG
Logistic	9.53%	9.64%	10.32%
SVM	10.29%	10.65%	10.01%

Fig. 3. Sample two digit image of 60x60 dimension

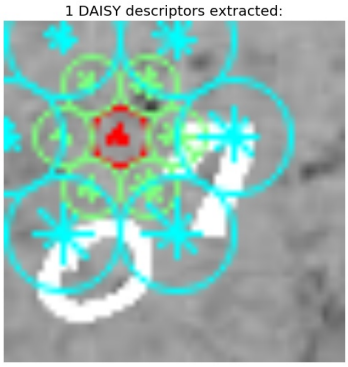

Fig. 4. Daisy Features

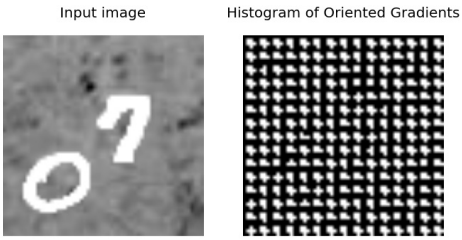

Fig. 5. Histogram of Oriented Gradients Features and Descriptors

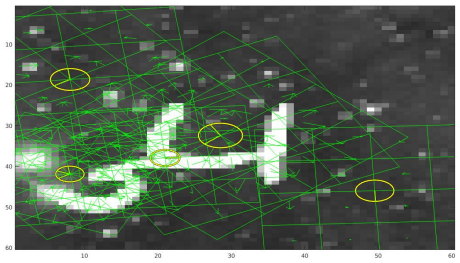

Fig. 6. SIFT Descriptors

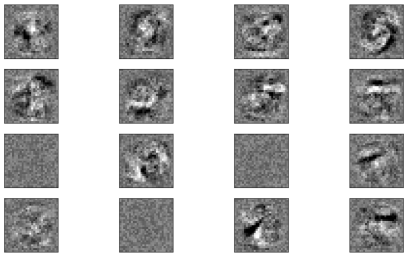

Fig. 7. Original MNIST filter visualization

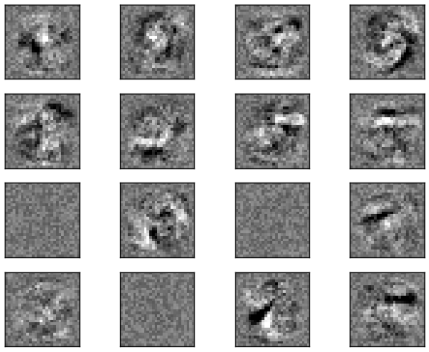

Fig. 8. Two digits filter visualization

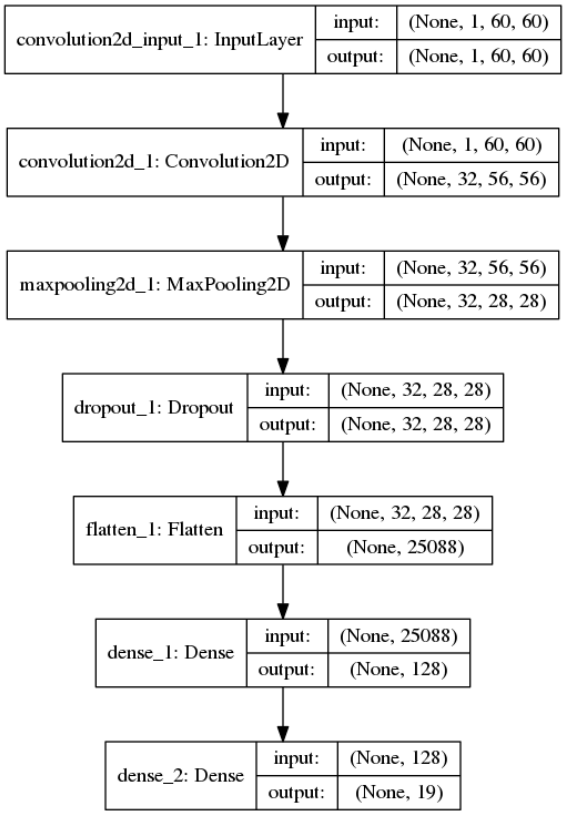

Fig. 9. ConvNet A

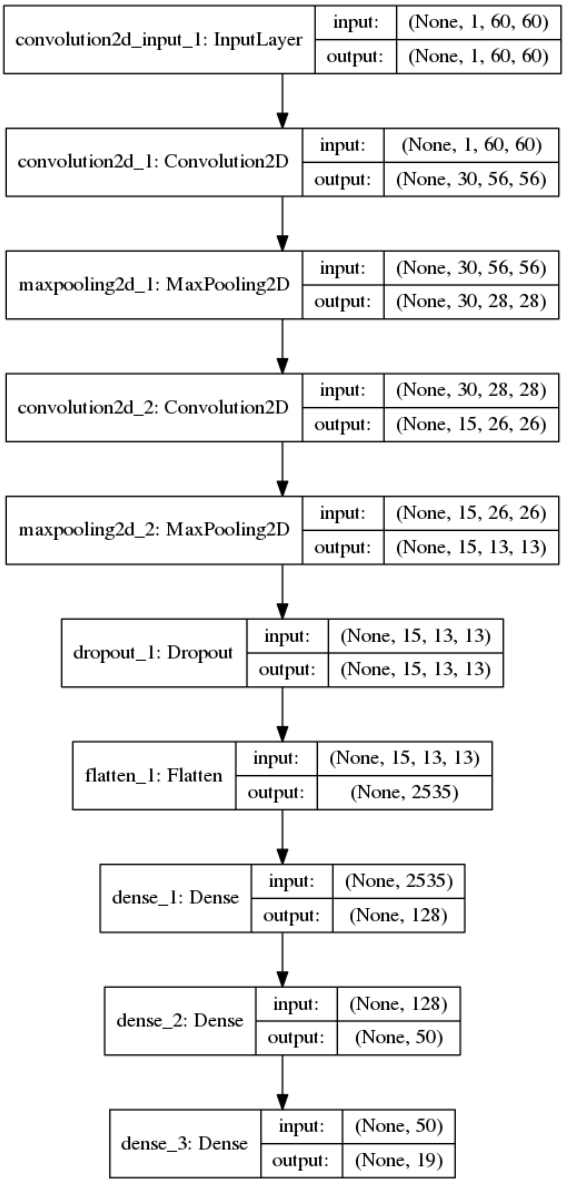

Fig. 10. ConvNet B

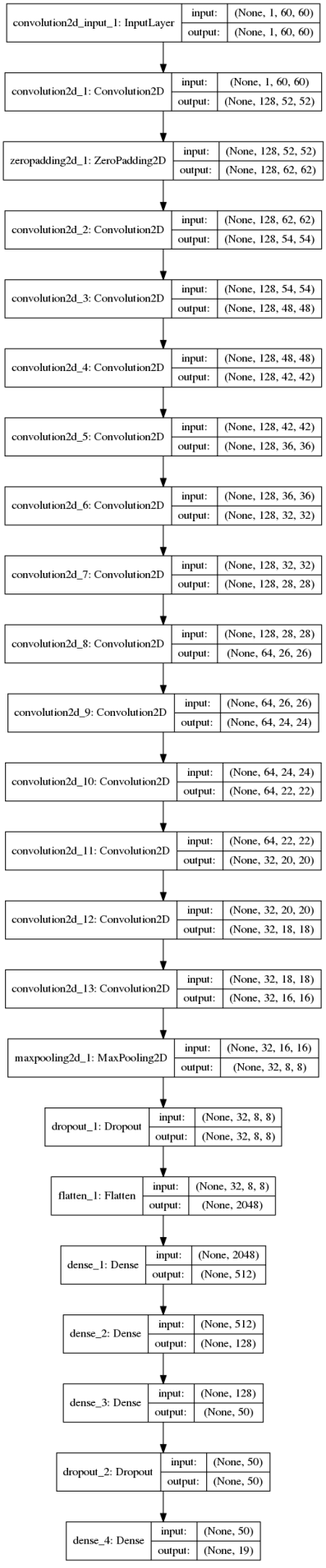

Fig. 11. ConvNet C

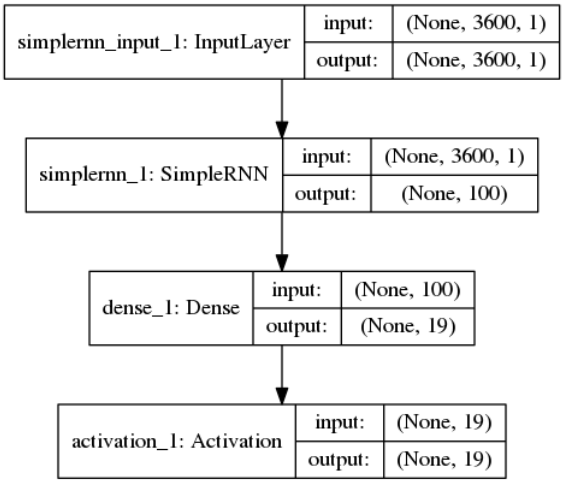
simplernn_input_1: InputLayer
input: (None, 3600, 1)
output: (None, 3600, 1)
simplernn_1: SimpleRNN
input: (None, 3600, 1)
output: (None, 100)
dense_1: Dense
input: (None, 100)
output: (None, 19)
activation_1: Activation
input: (None, 19)
output: (None, 19)

Fig. 12. RNN