COMP 551-001 Applied Machine Learning Modified MNIST - Team Naive Baes

(Project 4)

Frdric Ladouceur*, Joshua Lau†, Tiffany Wang‡
*Email: frederic.ladouceur@mail.mcgill.ca
Student ID: 260690491
†Email: joshua.lau@mail.mcgill.ca
Student ID: 260582760
‡Email: tiffany.wang@mail.mcgill.ca
Student ID: 260684152

***Abstract*—MNIST is a famous handwritten digit database in machine learning for digit classification [1]. In this project, the modified MNIST dataset is analyzed. The goal is to recognize the largest number in terms of pixel size. We show the outstanding performance of the deep convolutional neural network against basline linear earners and fully connected feed forward networks. We also demonstrate that in order to maximize prediction accuracy, careful image preprocessing, feature extraction and post-training ensemble methods are necessary.**

I. Introduction

The goal of this project is to design a computer vision algorithm to classify largest hand-written digits in an image. The dataset is a modified version of the MNIST dataset [1]. The challenge can broken down into two parts: 1. identifying the largest image, and 2. classifying the digit. Unlike the MNIST dataset, there are three digits per image and different textures are used in the background.

In the following report, we will discuss the approach we took to build the final training model. First, we will introduce the data, as well as the image preprocessing, using openCV, and normalization. Second, we will focus on the different models tested on the modified MNIST dataset, including the SVM, fully connected neural network and a deep convolutional net. Third, we will explain the different methodologies used to improve the prediction performance, such as regularization and ensemble methods. Last, we will discuss the accuracy results obtained from all models. We will focus on the advantages and the disadvantages of the deep convolutional network, as this the best suited model.

The algorithms in this project will be built using Scikit-Learn and Keras packages.

II. Feature Design

A. Dataset

Our algorithm is designed to perform image classification on a modified version of the well-known MNIST dataset [1]. Every example is formed from a 4096-bytes vector and a label. The vector describes a 1-channel 64x64 pixel image where the pixels are encoded as unsigned 8-bits words (grayscale from 0 to 255 in intensity). These images illustrate up to 3 MNIST digits randomly placed on an uneven gray background and randomly scaled to 40, 60, 80, 100 or 120 percent of the original digit size. The label represents the largest pixel-wise digit on the image.

B. Image Preprocessing

The idea behind the preprocessing is to get rid of the unimportant features. The features that describe the background convey no useful information for the classification. Therefore, they are set to zero, thus reducing the data's dimensionality (complexity). The pixels composing the digits have a high intensity; a simple threshold is set to dissociate them from the background. The preprocessing algorithm then proceeds to detect parts of the image that are not zero and draws bounding boxes around them. It performs such a task by using the OpenCV function *findContours()*. An example of this decoupage is illustrated on *figure 1*. We use the size of these boxes to determine the largest digits. The digits that have bounding squares larger or equal than 80 percent largest square dimension are kept, the others are set to zero. One might be tempted to use the size of the bounding boxes as a criterion to decide which digit is the largest. That would make of this problem just another simple MNIST digit classification one, which is fairly easy. However, we have too little information about how the dataset is created and that strategy could end up introducing labeling errors. Also, *findContours()* have some flaws. The algorithm mostly isolate individual digits, but sometimes, it draws contours around background remains or a pair of tightly-placed digits. We have to take precautions to avoid picking a double digit as the largest. We do that by neglecting bounding boxes that are too big in size. An example of this preprocessing is shown through figure 1 to 4.

Due to the size limitation of uploaded files, we have stored the processed image files on Amazon S3 storage. The submitted will directly be loading the data from the Amazon server.

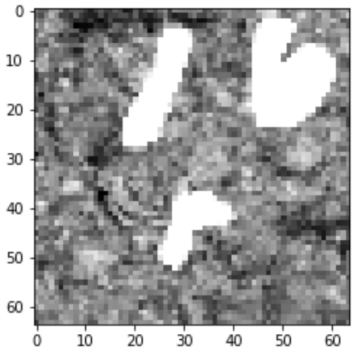

Fig. 1: Sample from provided modified MNIST dataset

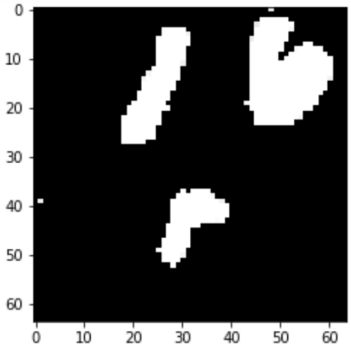

Fig. 2: Sample after background removal

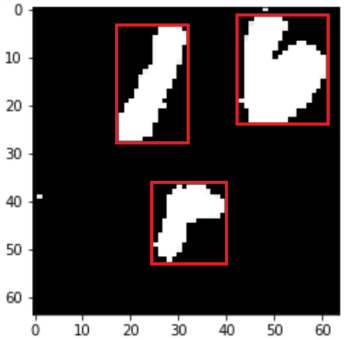

Fig. 3: Digit boundary box detection using OpenCV

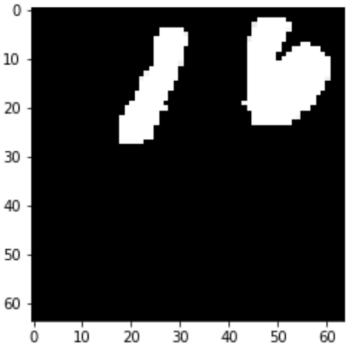

Fig. 4: Removal of smallest digit(s)

C. Image Normalization

After background removal as stated in the Image Preprocessing section, the images are re-scaled on a scale from 0 to 1. Essentially, the grayscale values are divided by 255.0. Then algorithm performs image normalization on the first layer on the each training batch using the following equation:

$$f(x) = \frac{x - \mu(x)}{\sigma(x)} \tag{1}$$

where:

- μ: is the mean of x
- σ: is the standard deviation of x

The idea is to have closer data values into order to weigh all features equally. The normalization is performed on each batch instead of the whole dataset because if the dataset should be normalized according to each training set for better performance. Normalizing the whole dataset would lose the effect of normalization. In terms of feature extraction, normalization eliminates outliers that may shift the features weights unevenly. Therefore, the contribution of each would be proportional according to distance, which obeys to the convolutional property. which has been found to allow for a faster convergence of the classifier when training [4].

III. Algorithms

A. Support Vector Machine

The intuition behind the support vector machine (SVM) is to construct a hyperplane or set of hyperplanes in a high- or infinite-dimensional space when the data is not linear in its dimensional space. A good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of all classes, also known as functional margin, as this would lower the generalization error of the classifier.

This baseline linear learner was chosen over logistic regression, as this is not a binary classification problem, the performance on which logistic regression is better known for. Therefore, the one-vs-rest training of SVM is more attractive and would intuitively render higher accuracy.

As the number of samples (50000) is much higher than the dimension (4096), the SVM model is trained with a Radial basis function (RBF) kernel.

Some hyper-parameters include the number of iterations, penalty and crammer_singer will be fine-tuned to improve the accuracy of the prediction. The model was fine-tuned using 5-fold cross-validation.

B. Fully Connected Deep Neural Network

The neural network was constructed by hand. In a fully-connected neural network, all inputs are activated and used to calculate the next layer. The features are utilized as nodes for the input layer, and there are 10 nodes (1 for each class) in the output layer. The tanh activation is used to calculate the activation of the hidden layers. It is a simple activation function that restricts the output between 0 and 1, and is centered at zero.

We chose to use the sigmoid activation at the output layer due to its property of ignoring the minimal structure of the dataset. It is also a generative model, which provides good representation between the model and the individual features and responses.

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

We used the logistic loss function to estimate the cost of the prediction

$$loss = -\frac{1}{n}\sum_{i=1}^{n} y - log(\hat{y}) + (1 - y) \cdot log(1 - \hat{y}) \tag{3}$$

A bias node provide flexibility, yet it does not always contribute into improving the performance. The model is tested with and without biases, and the addition of biases helped increase the accuracy by 0.2.

The feed-forward network is coded following the equations:

$$A = Input \times W + Bias \tag{4}$$

$$Output = activation(A) \tag{5}$$

The backward propagation network and weight updates are coded as followed:

$$W = W - \eta \cdot \frac{dW}{n} \tag{6}$$

$$Bias = Bias - \eta \cdot \frac{dBias}{n} \tag{7}$$

Where η is the learning rate.

To validate our model, we typically used a validation split rate of 0.05 and used 5-folds. We further fine-tuned this rate to find that 0.02 gives the best results. Multiple hyper-parameters can be tuned in the network: η, number of hidden layers, epochs, training and validation data set split.

C. Deep Convolution Network

Our final model is built as a deep convolutional neural network, further referred to as ConvNets. These networks are the most popular model in computer vision. Different from fully connected neural nets, ConVnets feed in the input to the next layer in small patches (locations). Their local connectivity allow better feature detections, which is important in image classification [6]. These features include edges, corners, features hard to detect in fully connected neural networks.

We chose to introduce multiple hidden layers within the network. Deep neural networks are more efficient in the representation of more complex functions using less parameters. Our network is built using a total of sixteen layers using normalization, ConvNet, pooling, dropout and fully connected layers.At the output of a ConvNet, a pooling layer performs a down sampling operation to reduce the variance of the model, hence reducing overfitting, and also acting as a filter to the image for better classification [6]. The idea is that where a feature is exactly located is not as important as the general location of that feature relative to the others. This pooling layer is then fed into a Dropout layer for regularization purposes, further discussed in the Methodology section. The final layers of the deep neural network are dense layers that classifies the output of the ConVnet. Our architecture follows that of VGGnet, as it has shown great performance on ImageNet Challenges [8].

Although the addition of more layers would help improving the performance, the additional computation-cost and slight accuracy improvement trade-off was not practical. Therefore, we decided to keep our network small.

The model can be found in the appendix.

IV. Methodology

A. Fine Tuning

1) Epoch and Batch sizes: Feeding in all 50,000 samples into the ConvNet is not ideal as it would be too computationally heavy. Training neural networks in smaller batch sizes helps reduce memory and reduce training time. However, smaller batches render less predictions, as the model does not have enough data to train the weights and biases [6].

The number of epochs and batch sizes are fine-tuned together, as the larger batches require less epochs. First, the batches are chosen to be power of 2, due to computers binary memories and for faster training convergences [7]. The model is then trained with batches of 32, 64 and 128 samples.

The number of epochs is tested from 10 to 25, and the learning trend of the model is used to determine the best epoch number. It is important to note that too many epochs may lead to overfitting, which is something we want to avoid.

2) Number of layers in the neural net: Some datasets do not require many hidden layers in the deep neural network, as it may be overkill. Therefore, the number of convolutional layers should also be tuned. Following the VGG ConvNet model, we had a base model with two ConvNet layers, followed by a pooling and a fully connected network layers as hidden layers. Our interest is to find the number of sets (consisting of two layers of ConvNets and one pooling layer) that can added onto our network to improve performance. We added a set of layers after each run to tune our model for the best performance all while taking into consideration the computation cost of the additional layers.

We found that having 3 sets of ConvNet layers render the best results. Our model then has a total of sixteen layers, as described in the Algorithm section.

B. Regularization

1) Dropout: Dropout is a regularization technique used to prevent overfitting [2]. When a layer of dropout is introduced, nodes are dropped (or deactivated) with probability p at random. At the end of a training stage, the removed nodes are added back into the network with their original weights (before the removal in the current training stage). This allows the weights to average out during training, and thus to prevent co-adaptation of units [2]. In addition, dropout is an efficient way to combine several different neural network

model architectures together while training, which is proven to improve model prediction accuracy [3].

2) Data Augmentation: Data Augmentation during training offers a more robust and accurate performance as it artificially provides more sample data [5]. These new images are essentially the original images, yet with noise. Therefore feeding these image into the network along with the original images, reinforces the network to learn how to detect specific features, and avoid overfitting. Image augmentation can be done by rotating, shifting, flipping, applying a filter to the image.

We introduced the augmented data using ImageDataGenerator in keras.preprocessing package. We fine-tuned the data augmentation with the following ranges.

- width_shift_range = 0.05 to 0.3
- height_shift_range = 0.1 to 0.3
- zoom_range = 0.08 to 0.1
- shear_range = 0.3
- rotation_range = 10 to 40

The best parameters are:

TABLE I: Best Hyper Parameter of Data Augmentation

Hyper-parameter	Value
width_shift_range	0.1
height_shift_range	0.1
zoom_range	0.08
shear_range	0.3
rotation_range	12

A test was conducted on an untuned deep ConvNet. The incorporation of the data augmentation helped increase the accuracy by 1.2%, from 94.4% to 95.6%.

3) ZCA whitening: This method is a linear transformation of the input features as a matrix into a matrix with new values whose covariance matrix is the identity matrix. The edited features/values in the new matrix are uncorrelated and have a variance value of 1 [11].

C. Bagging

Ensemble modeling combines multiple models to allow better predictive performance [3]. It is a technique used to prevent overfitting and increase the accuracy of machine learning algorithms used in statistical classification. When performing ensemble voting, the prediction of multiple models for a same input vote for the output they predict. The prediction that counts the most votes is chosen to be the output of this new meta-classifier. New training sets are generated by sampling from the training set with replacement. Models are fitted using the new training sets and are then combined by voting.

D. Decisions about training/validation split

The neural network model was tested on 1%-5% validation splitting on the input data.

V. Results

A. SVM

The support vector machine performance is not a good choice for the modified MNIST dataset. After fine-tuning, the best hyper-parameters were found to be one-vs-rest with a maximum iterations of 750. However, the validation accuracy is as low as random guessing, 0.143. Methods operating under the assumption of linear separability, in current or higher dimensional space, generally have a low accuracy in computer vision applications, as images are non linear. Furthermore, linear decision boundaries work better when there are little samples to separate compared to the dimensional space. However, as stated earlier, the dataset consists of 50000 samples against a dimension of 4096. Therefore, the low performance is expected.

B. Fully Connected Neural Network

The neural network model performance was not ideal with the modified MNIST dataset. With a validation split rates ranging from 0.02 to 0.3, the performance of the model stagnates at 0.11, which is random prediction rate.

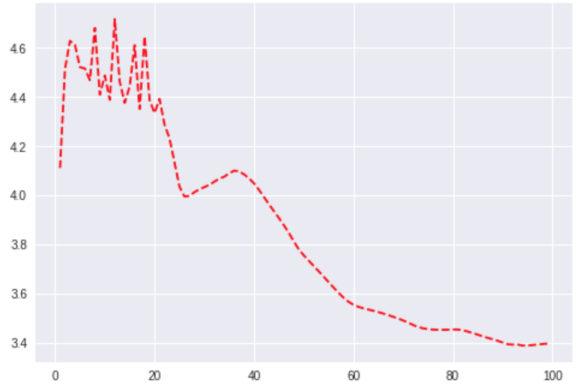

Fig. 5: Neural Network Training Loss Trend

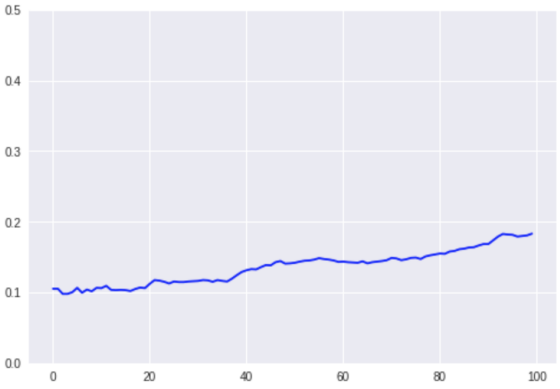

Fig. 6: Neural Network Training Accuracy Trend

As observed from the graphs above, we notice that validation accuracy stagnates around 0.10 regardless of the validation split, yet the loss keeps decreasing. This shows that the model

quickly overfits and is not suited for the project. This was an expected result as image analysis require careful extraction, which cannot be done within a fully connected neural network. In the next section, we will explore performance of the convolutional network to predict the modified MNIST dataset.

C. Deep Convolutional Network

Of the three models tested, ConvNet has the best performance without tuning with a validation accuracy of 91.4%. Upon the addition of several layers, we successfully obtained an accuracy of 95.6%.

We only introduced the ensemble method with the predictions outputs of this model, for its outstanding performance compared to SVM and a fully connected neural network. When bagging predicts with validation accuracies of 94.3%, 93.8%, 95.1% and 95.6%, we obtained an output with 96.3% accuracy.

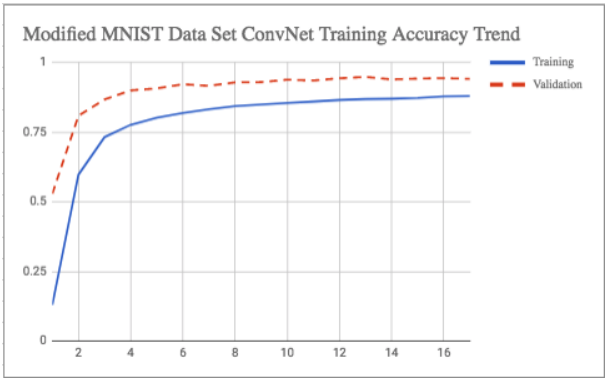

Fig. 7: Deep ConvNet Learning Trend graph - Accuracy

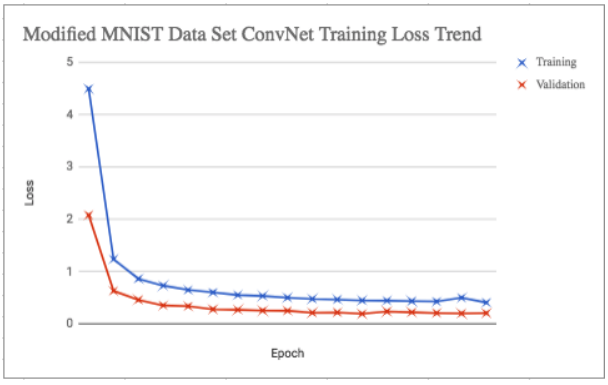

Fig. 8: Deep ConvNet Learning Trend graph - Loss

As observed in the graphs, after epoch 12, the loss and the accuracy stagnates. Training more epochs would lead over fitting. However, after fine-tuning the epochs from 10 to 25 on multiple slightly varied models, it is found that 17 epochs gave the best accuracy.

We added three layers of Dropout in our model. Typically, a dropout of 0.5 is found to best regularize a model. However, after fine-tuning the dropout rate from 0.2 to 0.6, it is found that p = 0.25 for the first two layers works best and increased the validation accuracy from 0.93 to 0.951. The last layer is however kept at p = 0.5.

TABLE II: Validation Accuracy per Dropout rate in ConvNet

Dropout rate	Validation Accuracy
0.2	0.945
0.25	0.951
0.3	0.943
0.35	0.938
0.4	0.942
0.45	0.934
0.5	0.932
0.55	0.933
0.6	0.924

We determined the best batch size for the model. Recall that the tested batch sizes were 32, 64, 128. The following results were obtained when training the same model through 17 epochs.

TABLE III: Validation Accuracy per batch size

Batch Size	Validation Accuracy
32	0.84
64	0.95
128	0.94

VI. Discussion

The Convolutional neural networks was best performing model, and the approach of our choice [10]. Therefore this section will be dedicated to discussing the advantages and the disadvantages of this algorithm.

A. Advantages of approach

The choice of ConvNet is advantageous against a regular fully connected neural network mainly for its local connectivity, allowing specific feature extraction. However, the addition of several regularization methods (ZCA, dropout, bagging, data augmentation) completes the model by reducing overfitting efficiently in different ways. The choice of using rectified linear units instead of the sigmoid function gives sparsity in the input matrix, helping in feature selection as well as lessening the amount of computations. Although we preprocessed the data, one of the main properties of a ConvNet is its capability of interpreting minimally processed image data, as ConvNet make the explicit assumption that the inputs are images. Therefore, the model should be highly effective with the original dataset as well.

B. Disadvantages of approach

In the ensemble approach (bagging), there may be a majority voting towards the right answer, as well as the wrong answer. This is because all the output were predicted from slightly varying models. Therefore, the weights are training similarly, and all models may output the same wrong

prediction for a specific sample.

When extracting the largest digit in the preprocessing process, we arbitrarily chose to keep all digits that is at least 80 percent of the largest bounding square. As mentioned previously, in some samples, two digits have the same largest bounding square width. Therefore the preprocessing image may have one or two digits. While this removes human error when making decisions, the images containing more than one digits creates an unbalanced dataset that may yields a lower accuracy.

C. Classifier performance

It is likely that the error on the test set originates from the incorrect classification of the numbers 6, 7, 9. As the numbers 6 and 9 are rotations of each other, and the features distinction between number 7 and 1, 2, 4, 9 is minimal.

D. Areas of future work

To potentially yield a higher accuracy, other ensembling approaches could be tested. Examples include the boosting or stacking approach. In the boosting approach, there are cases where the approach gives a better accuracy, but the approach is also more likely to overfit the data. In the stacking approach, the predictions of several learning algorithms are combined. We could also test the model against different gradient descent algorithms, such as Nadam, Adam, AMSGrad, AdaMax, etc.

When pre-processing the data, if the largest digits were scaled much larger fitting in whole image, the modified MNIST dataset essentially reverts back into an unmodified MNIST dataset. A more restrictive bounding box algorithm could be explored after further pre-processing (e.g. scaling) will allow for the creation of a more balanced dataset, which will in turn yield a higher accuracy.

An area to improve on is the understanding of different hyperparameters, in order to better fine-tune our model in the future.

VII. Statement of Contributions

1) Frederic Ladouceur: He was responsible of developing and coding the image preprocessing algorithms. Upon discussion with Tiffany, he decided on the background removal and largest digit extraction using OpenCV. As well, he assisted in building the training model, including the different regularization methods. Finally, he helped write the report on the parts that he contributed in, image preprocessing.

2) Joshua Lau: He developed the code for the SVM, and worked with Tiffany on the neural network code. Furthermore, he helped combining all the algorithms and methodologies in order to write the report.

3) Tiffany Wang: She mainly worked on the kaggle competition algorithm by developing the code of the ConvNet, fine-tuning, ensembling and doing the data analysis. She came together with Frederic to discuss different regularization methods. As well, she assisted Joshua in the coding of the neural network model. In addition, she helped writing the report on the parts that she contributed in.

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

VIII. References

References

[1] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition", Proceedings of the IEEE, 86(11):2278-2324, November 1998, Available: http://yann.lecun.com/exdb/mnist/

[2] Srivastava et al, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research 15*, p 1929-1958, 2014. Available: http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf

[3] Harris Drucker , Corinna Cortes , L. D. Jackel , Yann LeCun , and Vladimir Vapnik, "Boosting and Other Ensemble Methods" *Neural Computation*, 1994, 1289-1301

[4] Ioffe, Sergey and Szegedy, Christian, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", *JMLR: W&CP volume 37. Copy*, Proceedings of the 32^{nd} International Conference on Machine Learning, 2015

[5] Wang, Jason and Perez, Luis, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning", 2017

[6] Karpathy, Andrej, "CS231n Convolutional Neural Networks for Visual Recognition", Stanford University, 2017

[7] TensorFlow Optimizations on Modern Intel Architecture, "CIFAR-10 Classification using Intel Optimization for TensorFlow". Available at: https://software.intel.com/en-us/articles/cifar-10-classification-using-intel-optimization-for-tensorflow

[8] Simonyan, Karen and Zisserman, Andrew, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *ICLR*, 2015

[9] Nikita Joshi et al, "Improving Classification Accuracy Using Ensemble Learning Technique (Using Different Decision Trees)",*International Journal of Computer Science and Mobile Computing*, Vol.3 Issue.5, May-2014, pg. 727-732

[10] D.Ciresan, U.Meier, L.Gambardella "Convolutional Neural Network Committees for Handwritten Character Classification" DOI: 10.1109/ICDAR.2011.229 18-21 Sept. 2011 Available: http://ieeexplore.ieee.org/abstract/document/6065487/

[11] Kessy, Agnan, Lewin, Alex and Strimmer, Korbinian, 'Optimal whitening and decorrelation', *The American Statistician*, 2016

Layer (type)	Output Shape	Param #
lambda_2 (Lambda)	(None, 64, 64, 1)	0
conv2d_1 (Conv2D)	(None, 62, 62, 32)	320
conv2d_2 (Conv2D)	(None, 60, 60, 32)	9248
max_pooling2d_1 (MaxPooling2	(None, 30, 30, 32)	0
conv2d_3 (Conv2D)	(None, 28, 28, 64)	18496
conv2d_4 (Conv2D)	(None, 26, 26, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 13, 13, 64)	0
dropout_1 (Dropout)	(None, 13, 13, 64)	0
conv2d_5 (Conv2D)	(None, 11, 11, 128)	73856
conv2d_6 (Conv2D)	(None, 9, 9, 128)	147584
max_pooling2d_3 (MaxPooling2	(None, 4, 4, 128)	0
dropout_2 (Dropout)	(None, 4, 4, 128)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_1 (Dense)	(None, 512)	1049088
dropout_3 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5130

Total params: 1,340,650
Trainable params: 1,340,650
Non-trainable params: 0

Fig. 9: Model Summary