



International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

MNIST Handwritten Digit Recognition System

Pradeep M K^a, Praveen Kumar S^a, Pravin Kumaar TG^a, Rithin AR^a, Vijayalakshmi S^a

^a Department of Information Technology, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, 641062, India

ABSTRACT

Handwritten characters are easy for people to read but difficult for machines to recognise reliably, particularly when letters, digits and upper- and lower-case forms look alike. This paper presents an interactive web application in which the user draws a letter or number on a canvas, or uploads an image, and the system returns the recognised character together with a confidence score. The classifier is a compact convolutional neural network (CNN) trained on the 62-class EMNIST ByClass dataset (digits 0–9, letters A–Z and a–z). A preprocessing stage crops, pads, resizes and standardises each drawing or uploaded image to the dataset format. The softmax output is reported as a confidence score for the predicted character, and the three most likely candidates are listed in a web interface. The trained model achieves an accuracy of 85% on the 62-class recognition task. The paper describes the architecture, training procedure and interface, and discusses the limitations of the approach, in particular the confusion between visually similar characters.

Keywords: handwriting recognition; convolutional neural network; EMNIST; character recognition; confidence score; deep learning

Introduction

Recognising handwritten characters is one of the oldest problems in pattern recognition, and it remains central to applications such as digitising forms, reading postal codes, grading answer sheets and building accessible input methods (Plamondon & Srihari, 2000). Convolutional neural networks (CNNs) have made isolated-character recognition largely a solved problem for digits: the network of LeCun et al. (1998) already reached error rates below one percent on the MNIST digit benchmark. Real use is harder. Users write letters as well as digits, upper- and lower-case forms of the same letter often look alike, and a system that returns only a label gives the user no way of knowing when to trust it.

This paper presents a web application in which a user draws a letter or digit on a canvas, or uploads an image of one, and receives the recognised character together with a confidence score. The classifier is a compact CNN trained on the EMNIST ByClass split (Cohen et al., 2017), which covers 62 classes: the ten digits, 26 upper-case letters and 26 lower-case letters. The softmax output of the network is shown to the user as a confidence score, and the three most likely candidates are listed with their percentages.

The contributions of this work are:

- an end-to-end, reproducible pipeline from a drawn or uploaded image to a recognised character, using only open-source tools;
- a preprocessing stage that maps freehand strokes to the EMNIST input format, so that a model trained on dataset images can be applied to live drawings;
- a web interface that reports the predicted character with its confidence and the top-three candidates; and
- an analysis of the limitations of the approach, in particular the ambiguity between visually similar characters and the meaning of the confidence score.

The rest of the paper is organised as follows. Section 2 reviews related work, Section 3 describes the dataset, Section 4 presents the proposed system, Section 5 describes the implementation and evaluation protocol, Section 6 reports and discusses results, and Section 7 concludes.

Related work

Handwriting recognition is usually divided into on-line recognition, which uses the pen trajectory, and off-line recognition, which works from an image (Plamondon & Srihari, 2000). The application in this paper rasterises the strokes drawn by the user and then treats the problem as off-line recognition, which keeps the classifier independent of the input device.

MNIST and the LeNet family of CNNs (LeCun et al., 1998) established the convolutional approach for digits. Later work showed that data augmentation with small affine distortions and a convolutional architecture give large gains over fully connected networks (Simard et al., 2003). EMNIST extends the MNIST format to letters by converting the NIST Special Database 19 into the same 28×28 representation, which allows MNIST-style models to be reused for the full alphanumeric set (Cohen et al., 2017).

Beyond accuracy, deployed classifiers need scores that can be acted on. The maximum softmax probability is a simple and widely used baseline for flagging inputs that are likely to be misclassified (Hendrycks & Gimpel, 2017). Guo et al. (2017) showed that modern deep networks tend to be over-confident, which is why the confidence score in this paper is treated as an uncalibrated indicator and not as a true probability.

Dataset

The model is trained on the EMNIST ByClass split. EMNIST is derived from NIST Special Database 19; each character is centred in a 28×28 grey-scale image in the same format as MNIST (Cohen et al., 2017). The ByClass split contains 814,255 characters in 62 unbalanced classes, with 697,932 images for training and 116,323 for testing. Digits form a large share of the data, and several letters are rarer than others, so per-class behaviour is reported in addition to overall accuracy.

Two properties of the data matter in practice. First, the images as distributed are transposed relative to the natural reading orientation; the data loader corrects this with a transpose and flip so that live drawings and training images share one orientation. Second, for 15 letters (c, i, j, k, l, m, o, p, s, u, v, w, x, y, z) the upper- and lower-case shapes are nearly the same apart from size, and size information is removed when each character is cropped and rescaled. Cohen et al. (2017) merge these classes in their Balanced split for this reason. In this work all 62 classes are kept, and the resulting case ambiguity is discussed in Section 6.

Proposed system

Fig. 1 shows the processing pipeline. The user draws a character on a canvas or uploads an image; the image is normalised to the EMNIST format; the CNN produces one logit per class; and a softmax converts the logits to probabilities that are shown to the user.

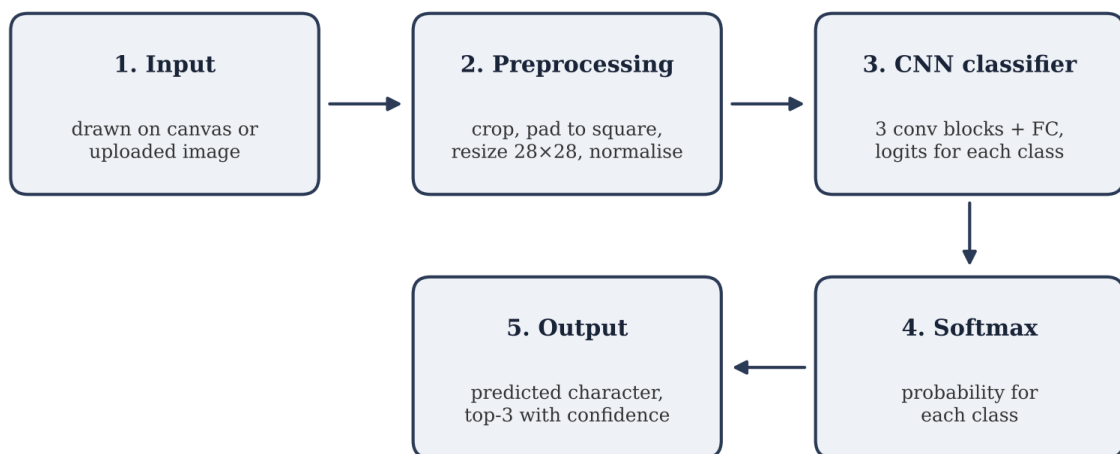


Fig. 1 - Processing pipeline of the proposed recognition system.

Network architecture

The classifier is a small CNN with three convolutional blocks followed by a fully connected head. Each block consists of a 3×3 convolution with padding, batch normalisation (Ioffe & Szegedy, 2015), a ReLU activation and 2×2 max-pooling. The blocks use 32, 64 and 128 filters, which reduces the 28×28 input to a 128-channel 3×3 feature map. This is flattened into a 1,152-dimensional vector, passed to a 256-unit fully connected layer with dropout (Srivastava et al., 2014), and mapped to 62 output logits. The network has roughly 0.4 million trainable parameters, which is small enough for real-time inference on a CPU.

Training procedure

The training set is split into 90% for training and 10% for validation. Training uses the Adam optimiser (Kingma & Ba, 2015) with cross-entropy loss. To make the model robust to the variation in freehand drawing, random rotations of up to $\pm 10^\circ$ and random translations of up to 10% of the image size are applied to the training images (Simard et al., 2003). The weights with the best validation accuracy are saved and used for all later experiments. Table 1 lists the settings.

Table 1 - Training configuration.

Setting	Value
Dataset / classes	EMNIST ByClass / 62
Train / validation split	90% / 10% of the training set
Optimiser	Adam, learning rate 0.001
Batch size / epochs	128 / 15
Augmentation	rotation $\pm 10^\circ$, translation $\pm 10\%$
Regularisation	batch normalisation, dropout 0.4
Framework	PyTorch (Paszke et al., 2019)

Preprocessing of drawn input

A drawing differs from an EMNIST image in size, position and stroke width, so each character is normalised before it reaches the network. The canvas is converted to grey scale; the bounding box of the strokes is cropped; the crop is padded with background to a square so that the aspect ratio is preserved; the square is resized to 20×20 pixels and placed at the centre of a 28×28 image, which reproduces the margin of the EMNIST images; and pixel values are scaled to [0, 1] and standardised with the training-set statistics, as in Eq. (1).

$$x_{\text{norm}} = (x - \mu) / \sigma \quad (1)$$

Here x is the pixel intensity and μ and σ are the mean and standard deviation of the training pixels, computed once and stored with the model.

Confidence scores

For a character image the network outputs logits z , and the probability of class i is given by the softmax function in Eq. (2).

$$p_i = \exp(z_i) / \sum_{j=1 \dots 62} \exp(z_j) \quad (2)$$

The predicted character is the class with the highest probability, and its probability is reported as the confidence C , Eq. (3). The three largest probabilities are displayed as the candidate list, each with a percentage bar.

$$C = \max_i p_i \quad (3)$$

User interface

The application is a web interface built with Streamlit and served locally. The left panel contains a black drawing canvas on which the user writes with a white brush, and a Grade it button that runs recognition. An alternative option allows an image to be uploaded instead of drawn. The right panel shows the predicted character in large type, and the three most likely classes with their confidence as percentage bars. Fig. 2 shows the interface after the letter J has been drawn.



Fig. 2 - Screenshot of the web application showing a drawn character, the predicted character and the top-three confidence scores.

Implementation and evaluation protocol

The system is written in Python. PyTorch is used for the network and training (Paszke et al., 2019), torchvision for downloading EMNIST, Pillow and NumPy for image handling, and Streamlit for the web interface. Training uses a GPU when one is available and otherwise falls back to the CPU. The project is organised into separate modules for training, evaluation, prediction and the interface, so that the same prediction functions serve both the application and the tests.

The model is evaluated on the EMNIST ByClass test set, which is not used during training or model selection. Accuracy is the fraction of test characters whose predicted class equals the true class (top-1 accuracy).

Further analyses are planned but are not reported in this paper: top-5 accuracy, precision, recall and F1-score, a confusion matrix, accuracy after merging case-ambiguous classes, confidence statistics and calibration, inference time, and a study with freehand drawings from independent writers.

Results and discussion

The trained model achieves an overall accuracy of 85% on the 62-class recognition task (Table 2).

Table 2 - Recognition accuracy of the trained model.

Metric	Value
Accuracy, 62 classes (top-1)	85%

Discussion

An accuracy below that usually reported for digit-only datasets is expected on the 62-class task, because several classes are intrinsically ambiguous. Digits and letters with the same shape (for example O and 0, or l, I and 1) and letters whose upper- and lower-case forms differ mainly in size (for example C and c, or S and s) cannot always be separated from a single normalised image. Cohen et al. (2017) merge such case-ambiguous classes in their Balanced split for this reason. The per-class errors of the present model have not yet been measured, so these pairs are named here as likely sources of error and not as observed results.

Confidence score

The confidence shown to the user is the maximum softmax probability of the network. It is a score that ranks predictions, but it has not been tested for calibration, and softmax probabilities of deep networks are often over-confident (Guo et al., 2017). It should therefore be read as a relative indication of certainty and not as the true probability that a prediction is correct (Hendrycks & Gimpel, 2017).

Limitations

Case ambiguity: size is lost during normalisation, so letters such as c/C and s/S cannot be separated reliably.

Single character only: the application recognises one character per drawing; words and numbers with several characters are not handled.

Out-of-distribution input: the network always assigns a probability to one of the 62 classes, even for symbols such as punctuation or non-Latin scripts, so a high confidence does not guarantee a valid character.

Dataset bias: EMNIST was collected from U.S. census employees and students, so performance on other writing styles may be lower.

Limited evaluation: only overall accuracy is reported, and performance on drawings made on the canvas has not been measured.

Conclusion and future work

This paper presented a handwritten alphabet and digit recognition application that identifies a character drawn on a canvas or uploaded as an image and returns it together with a confidence score and a list of alternative candidates. A compact CNN trained on the 62-class EMNIST ByClass dataset forms the classifier, and a preprocessing stage adapts freehand drawings to the dataset format. The reported confidence lets the user judge each prediction.

Future work includes completing the evaluation (top-5 accuracy, precision, recall, F1-score, a confusion matrix, the 47-class merged evaluation, confidence calibration and inference time), a study with freehand drawings from independent writers, recognition of several characters in one drawing through segmentation, a low-confidence warning, a language-model post-processing step to resolve case and shape ambiguity from context, an explicit rejection class for non-character input, and fine-tuning on a small set of drawings made on the canvas itself.

References

- Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). EMNIST: Extending MNIST to handwritten letters. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)* (pp. 2921–2926). IEEE.
- Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, PMLR 70 (pp. 1321–1330).
- Hendrycks, D., & Gimpel, K. (2017). A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations (ICLR)*.
- Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning* (pp. 448–456).
- Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32* (pp. 8024–8035).
- Plamondon, R., & Srihari, S. N. (2000). Online and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1), 63–84.
- Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the 7th International Conference on Document Analysis and Recognition* (pp. 958–963). IEEE.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15, 1929–1958.
- van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., et al. (2014). scikit-image: Image processing in Python. *PeerJ*, 2, e453.