DeepSign: Efficient Siamese Convolutional Neural Networks for Signature Verification

Jonathan X Wang
Stanford University
Biomedical Informatics
jon.wang@ucsf.edu

Kevin Ko
Stanford University
Computer Science
kevinko@cs.stanford.edu

Abstract

Signatures are a common form of verification and a target for fraud. Current state of the art verification algorithms involve siamese convolutional neural networks. Here, we apply siamese convolutional neural networks that are both efficient and high performing for the task of signature verification. We created our own SqueezeNet-inspired efficient architecture, DeepSign, that uses 65% fewer parameters than Google's MobileNetv2 and 97% fewer parameters than the current state of the art, SigNet. We test our models on both the CEDAR and BHSig260 datasets and demonstrate that our model outperforms both models in all evaluation metrics (accuracy: 0.85, precision: 0.76, recall: 0.84, AUROC: 0.93). This lightweight model is readily applicable to mobile devices for both online or offline signature verification.

1. Introduction

Signatures are widely used as verification for checks, contracts, and prescription drugs. This makes signatures a common target for fraud. Manual fraud detection conducted by human experts is not only time consuming, but expensive due to the volume of signatures and the cost to hire an expert inspector. Due to recent advances in facial identification verification algorithms, we are interested in using similar algorithms for mobile automated signature verification. The following study explores the potential for an automated, open-source, and efficient deep learning model for signature verification.

1.1. Problem Statement

The input to our algorithm is a pair of signatures $s1_i$ and $s2_i$. We then use a Siamese neural network to output whether these signatures are from the same person or that one of them is forged.

2. Related Work

Work in the field of automated signature verification had its first major breakthroughs in the 1980s [1]. These methods generally involve labor intensive preprocessing and feature extraction on signatures which were then compared using a similarity metric [2, 3, 4]. In 2004, the first international signature verification task was announced [5], marking the beginning of larger collections of datasets like CEDAR and BHSig260 for public use [6, 7]. More recently, advances in computing power and databases for image classification like ImageNet have resulted in algorithms known as neural networks to gain popularity due to their high performance without feature extraction [8, 9]. In 2012, AlexNet demonstrated the ability of convolutional neural networks (CNNs) to to generate state of the art results on ImageNet classification [8]. More recently, an study known as SigNet demonstrated that convolutional siamese networks outperform all previous work in signature verification tasks [10]. For this reason, we focus our baseline and experimental models on the siamese network architecture.

The siamese neural network architecture, originally proposed in 1994, consists of two identical neural networks that share the same parameters and weights [11]. During training, these two networks share the same parameter updates. In addition, these networks have been shown to generate state of the art performance for facial recognition tasks as well [12].

Recent work has been done to develop models that achieve similar classification results on ImageNet, but with fewer parameters. The effort here is to develop models for faster evaluation with less memory and hard drive space. This is particularly applicable for real-time offline mobile signature verification on mobile phones or areas with poor network access. One recently developed mobile model is known as SqueezeNet, which is cited to achieve performance similar to AlexNet but with 50x fewer parameters and 0.5MB model size. Additionally, this architecture has been shown to perform well on facial verification tasks,

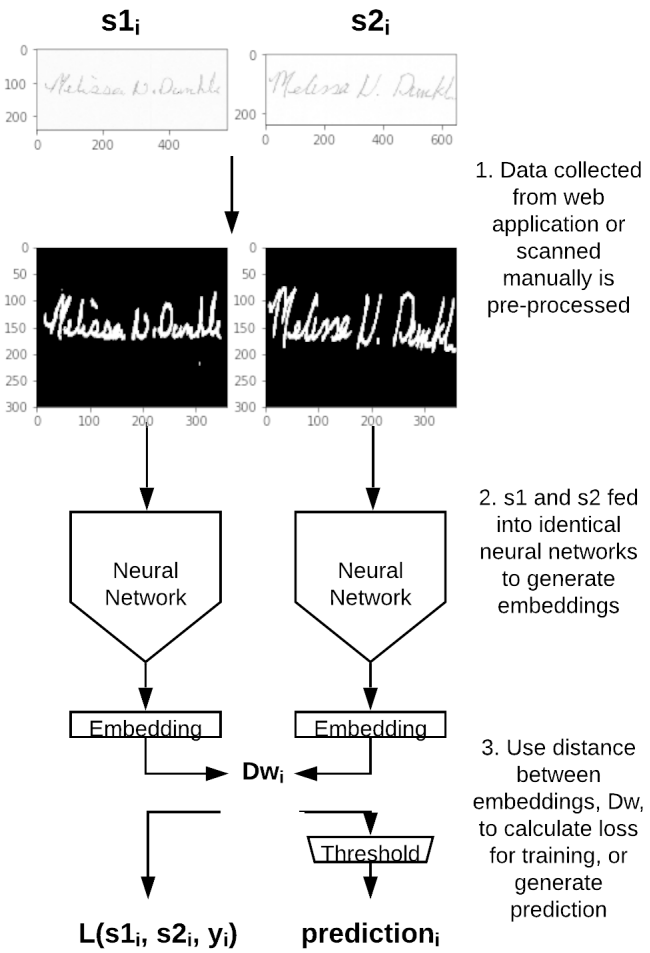

Figure 1. Overview of technical approach using a siamese neural network. First, we pre-process inputs gathered from scanned signatures or a mobile/web application. Secondly, we feed these two inputs into two of the same models that generates embedding representations of these two images. Finally, we use the distance between these two embeddings as an input into a contrastive loss function for training. After training, this distance can also be used to generate a prediction about whether the pair is forged or from the same person.

which points towards a potential application in signature verification as well [13, 14].

The most recent state of the art in mobile models is MobileNetv2, developed by Google [15]. MobileNetv2 features an inverted residual structure with shortcut connections between bottleneck layers, an intermediate expansion layer using depthwise convolutions, and the removal non-linearities in narrow layers to maintain representational power. Additionaly, this model has pretrained weights on ImageNet readily available for public use. A variety of recent work has demonstrated that transfer learning from pretrained weights can lead to high performance on smaller datasets such as ours [16, 17].

3. Methods

We use Tensorflow 2.0a, scikit-learn 0.21, OpenCV 4.1, Matplotlib 3.1, and Python 3.6 frameworks for this project [18, 19, 20, 21, 22]. We base our baseline and experimental models off a recent study SigNet, which demonstrated that convolutional siamese networks outperform all previous work in signature verification tasks [10]. Figure 1 shows an overview of our approach for this task.

3.1. Siamese Neural Network Architecture

The siamese neural network architecture consists of two identical neural networks that share the same parameters and weights during training, but recieve two different inputs [11]. The last layer of the two identical networks consists of a dense layer, which create two vector outputs of the same dimensionality. The two outputs are joined by a contrastive loss function [23] that computes a similarity metric between the scores such that forged-original pairings lead to a higher loss while original-original pairings lead to lower loss (1).

$$L(s1_i, s2_i, y_i) = (1-y_i)Dw_i^2 + y_i \max(0, m - Dw_i)^2 \quad (1)$$

Here, $s1_i$ and $s2_i$ are the ith pairing of signatures; y_i is a binary variable that designates whether pairing i consists of original-original (1) or forged-original (0) pair; m is the margin, which is equal to 1 in our case. D_w is the Euclidean distance between the two outputs of the shared sub-model.

This unique loss function leads to the generation of an embedding space where vector outputs of the siamese networks that are closer in Euclidean distance are more likely to be original-original pairings.

3.2. Baselines

Our baseline model is SigNet, which has state of the art performance on the datasets we have acquired [10]. SigNet is a convolutional siamese network with identical neural networks that are constructed as outlined in Table 1. Preprocessing is handled identically to the paper, except we binarize our data rather than normalize and perform an additional smoothing step. We do not anticipate these changes to cause any significant differences in network performance. We use the same hyperparameters reported in the paper. The only modification is we used early stopping when validation accuracy began decreasing. For more details on the architecture, see the original SigNet paper [10].

Layer	Parameters	Output Shape
Conv2D	$k = 11$, $s = 4$	(37, 53, 96)
BatchNorm2D	$\epsilon = 10^{-6}$, $\rho = 0.9$	(37, 53, 96)
MaxPool2D	$k = 3$, $s = 2$	(18, 26, 96)
ZeroPadding2D	pad $= 2$	(22, 30, 96)
Conv2D	$k = 5$, $s = 1$	(18, 26, 256)
BatchNorm2D	$\epsilon = 10^{-6}$, $\rho = 0.9$	(18, 26, 256)
MaxPool2D	$k = 3$, $s = 2$	(8, 12, 256)
Dropout	$p = 0.3$	(8, 12, 256)
ZeroPadding2D	pad $= 1$	(10, 14, 256)
Conv2D	$k = 3$, $s = 1$	(8, 12, 384)
ZeroPadding2D	pad $= 1$	(10, 14, 384)
Conv2D	$k = 3$, $s = 1$	(8, 12, 256)
MaxPool2D	$k = 3$, $s = 2$	(3, 5, 256)
Dropout	$p = 0.3$	(3, 5, 256)
Flatten	-	(3840,)
Dense	kernel L2 $= .0005$	(1024,)
Dropout	$p = 0.5$	(1024,)
Dense	kernel L2 $= .0005$	(128,)

Table 1. SigNet Architecture

3.3. Experimental Models

The goal of the following experimental models is to find a model with lower number of parameters and model storage size for offline mobile applications of signature verification.

Our first experimental model involves transfer learning from a MobileNetv2 pretrained on ImageNet [15]. A variety of recent work has demonstrated that transfer learning can lead to high performance on smaller datasets such as ours [16, 17]. MobileNetv2 uses a unique inverted residual block, known as bottleneck, which involves an intermediate expansion layer that incorporates depthwise convolutions to filter features. Additionally, it removes non-linearities in the lower dimensional layers to prevent the loss of representational power. In specific, these blocks are composed of a convolutional layer using 1×1 kernel and ReLU activation, a depthwise convolutional filter using a 3×3 kernel and ReLU activation, and finally a convolutional linear using a 1×1 kernel and linear activation. We remove the top layer of the MobileNetv2 and freeze all the weights in the original model. One problem we encountered is that the pre-trained models require a 3-channel input, and since our images are grey-scaled they start with only a single channel. To fix this issue, we run our input through a 10 filter 1×1 kernel convolutional layer and a 3 filter 1×1 kernel convolutional layer, both of which preserve the image size and use ReLU activations. This has been shown to be an effective way to convert 1-channel to 3-channel for use in pretrained models [24]. Finally we add a global average pooling layer, dense layer, dropout layer, dense layer, and an L2 normalization layer to produce the outputs. The overall structure of the adapted MobileNetv2 architecture is outlined in Table 2.

Layer	Parameters	Output Shape
Conv2D	$k = 1$, $s = 1$	(224, 224, 10)
Conv2D	$k = 1$, $s = 1$	(224, 224, 3)
MobileNetv2	-	(7, 7, 1280)
GlobalAvgPool2D	-	(1280,)
Dense	kernel L2 $= .00047$	(512,)
Dropout	$p = 0.24$	(512,)
Dense	kernel L2 $= .00047$	(128,)
L2 Norm	-	(128,)

Table 2. Adapted MobileNetv2 Architecture

Our second experimental model named DeepSign is adapted from SqueezeNet, in particular, it exploits the use of Fire blocks. Fire blocks are composed of a convolutional layer using 1×1 kernel followed by two branching convolutional layers with 1×1 and 3×3 kernels. The last layer in the Fire block concatenates the two results. The **sqz** parameter in the Fire block indicates the number of filters for initial layer in the block. Additionally, the **exp** parameters dictates the number of filters for the branching convolutional layers. We propose a shallower version of the SqueezeNet model that overall has a fewer number of filters and includes dropout layers for regularization. In the top-most block of our DeepSign model we opt out of using traditional fully connected layers and instead use a global average pooling layer to output the spatial average of the feature maps from

the last convolutional layer. Doing so reduces the number of trainable parameters vastly and also reduces issues of overfitting that apply to fully connected layers [25]. The overall structure of the adapted DeepSign architecture is outlined in Table 3.

Layer	Parameters	Output Shape
Conv2D	$k = 3, s = 2$	(149, 179, 64)
MaxPool2D	$k = 3, s = 2$	(74, 89, 64)
Fire	sqz $= 16$, exp $= 64$	(74, 89, 128)
Fire	sqz $= 16$, exp $= 64$	(74, 89, 128)
MaxPool2D	$k = 3, s = 2$	(36, 44, 128)
Dropout	$p = 0.2$	(36, 44, 128)
Fire	sqz $= 32$, exp $= 72$	(36, 44, 144)
Fire	sqz $= 32$, exp $= 96$	(36, 44, 192)
Fire	sqz $= 32$, exp $= 128$	(36, 44, 256)
MaxPool2D	$k = 3, s = 2$	(17, 21, 256)
Dropout	$p = 0.2$	(17, 21, 256)
Conv2D	$k = 1, s = 1$	(17, 21, 256)
GlobalAvgPool2D	-	(256,)
Dense	-	(128,)
L2 Norm	-	(128,)

Table 3. DeepSign Architecture

Table 4 summarizes the reduced model complexity that both DeepSign and MobileNetv2 offer.

Model	Parameters	Storage Size (MB)
SigNet	6,460,974	25
MobileNetv2	721,589	12
DeepSign	**256,992**	**1.1**

Table 4. DeepSign and MobileNetv2 demonstrate advantages in both number of trainable parameters of the models and model storage size.

4. Dataset

We use the following datasets for our task:

- CEDAR (America): 55 persons, 24 genuine signatures per person, 24 forged signatures per person (https://cedar.buffalo.edu/handwriting/HRdatabase.html) [6]
- BHSig260 (Bengali): 260 persons, 24 genuine signatures per person, 30 forged signatures per person (https://goo.gl/9QfByd) [7]

We generate our train/validation/test sets by performing a 60/20/20 random split by person. Since our data consists of pairs of signatures, we make all possible unique combinations from the signatures. Specifically, we create negative examples, namely forged-original pairs, by taking every forged signature for a person and pairing them with that person's original signatures. In CEDAR's case, there would be $24 \times 24 = 576$ forged-original pairs for every person. To create the positive examples, namely original-original pairs, we take every original signature for a person and pair it with the remaining person's original signatures. For the CEDAR dataset, there would be $\sum_{i=1}^{24} i = 300$ original-original pairs for every person.

We also used a combined dataset composed of both CEDAR and BHSig260.

Table 5 summarizes the total number of examples we had in our train/validation/test sets per dataset.

Dataset	Train	Validation	Test
CEDAR	$28,116$	$9,372$	$9,372$
BHSig260	$180,180$	$60,060$	$60,060$
Combined	$208,296$	$69,432$	$69,432$

Table 5. Dataset Splits

4.1. Pre-processing

4.1.1 CEDAR

First, we grayscale all images. Next, we apply a binary threshold to clamp all pixel values that are ≥ 225 at 255, and all other values at 0. Because of the handwritten nature of the CEDAR dataset, there is a lot of noise on the edges of the signature. To remove this noise, we apply a median blur with kernel size 3 for 15 iterations, and then apply a binary threshold at 210. Lastly, the image is resized using binary interpolation to the desired height and width and is inverted such that the background and signature are, respectively, 0 and 1. For SigNet, we resize to 155×220 as specified in the original paper. For MobileNetv2, we resize to 224×224 as specified by the pretrained MobileNetv2 model in Tensorflow Keras. Finally, for DeepSign, we resize the image to 300.

4.1.2 BHSig260

This dataset was originally grey scaled and binarized. Thus, we simply resized the image through binary interpolation and inverted the image such that the background and signature are, respectively, 0 and 1. We resize according to dimensions specified above in CEDAR data pre-processing.

5. Experimental details

5.1. Training parameters

For the SigNet baseline, we used the same hyperparameters from the original paper, except for the number of epochs. We performed early stopping as soon as the validation accuracy began to decrease, which occured at 3 epochs.

Even using these lower complexity models, each epoch still required around 10-20 minutes on a NVIDIA Tesla-V100 due to our large dataset size. This made it difficult to perform hyperparameter search. For MobileNetv2 and DeepSign, we choose to use the Adam optimizer as it has been shown to have less variance in performance based on other hyperparameters [26]. Additionally, we chose to use a batch size of 32 on both MobileNetv2 and DeepSign due its faster training time per epoch.

For the MobileNetv2 architecture, we found that the default learning rate of 0.001 generated a very high validation accuracy of 0.97. We train the model for 5 epochs because there was a noticeable plateau in validation accuracy after 5 epochs. After fixing the number of epochs and the learning rate, we tuned the model for 8 hours on randomly sampled values of the l2 regularization from a logarithmic scale of 0.0001 to 0.01 and dropout rate from a uniform scale of 0 to 0.4. Ultimately, the best values found were 0.00047 and 0.24 respectively.

For the DeepSign architecture, we mainly focused on the squeeze and expand parameters in the Fire blocks. We initially encountered the model overfitting to the training data, which lead to poor performance on the validation data. We reduced the number of filters used in the convolutional layers and decreased the depth of the network and then tuned the dropout probabilities to 0.2 to decrease variance and then number of parameters. The biggest performance gain was achieved by reducing the learning rate to 0.00003.

5.2. Evaluation

We report accuracy, precision, recall and AUROC for our evaluation metrics. Accuracy is calculated using a threshold on the euclidean distance between $s1_i$ and $s2_i$, such that any distance above the threshold is considered a forged-original pair, while any pair below the threshold is considered a original-original pair. These predictions are then compared to their true identity to generate an accuracy score. This threshold is chosen by generating a plot of the accuracy against threshold on the validation set and choosing the threshold which yields the maximum accuracy on the validation set. Thus, each model has its own unique threshold (i.e. 0.30 for SigNet and 0.70 for SqueezNet FILL THIS IN), which is applied to the test set to generate a final accuracy metric. These same thresholded values are used to calculate both precision and recall as well, which are standard metrics for how relevant and how complete the predicted original-original pairings are. AUROC is calculated using standard metrics and represents the overall ability of the model to discriminate between forged-original and original-original pairs.

We plot an ROC-curve to visualize the tradeoff between sensitivity and specificity between our model. We also perform a qualitative analysis on 9 high, middle, and low distances between pairs outputed by DeepSign.

Initially, we train on all datasets combined and test the model on the individual datasets as well as the combined dataset. To analyze the generalizability of our model to new languages and datasets, we also train our models on the CEDAR dataset and test them on both the CEDAR and BHSig260 datasets.

6. Results and Discussion

We first train our model on the entire dataset, and generate evaluation metrics for testing on the CEDAR, BHSig260, and combined dataset (Table 6). Our DeepSign model outperforms both SigNet and MobileNetv2 in accuracy for both individual datasets and overall. This is particularly surprising, as we expected the MobileNetv2 transfer-learned from ImageNet to perform best. We suspect our prediction task may be very different from the original ImageNet task, and that fully training the MobileNetv2 or unfreezing some layers may result in better performance. Additionally, DeepSign outperforms on all metrics for the BHSig260 and overall dataset. On the CEDAR dataset, however, it only performs best on accuracy. This suggests that the DeepSign model may be overfitting to the BHSig260 dataset, as it makes up a larger proportion of the training data. This led us to question the generalizability of the model to different datasets or languages, as it seemed to be more prone to overfitting to the BHSig260 dataset.

To investigate the generalizability of the model, we performed an experiment where we trained a model on the CEDAR (American) and tested it on both the CEDAR and BHSig260 (Bengali) datasets (Table 7). There are two interesting findings that we had from this experiment. First, after training specifically on the CEDAR dataset rather than the combined dataset (as in Table 6), DeepSign outperforms the other two models in all metrics except precision, while previously when trained on the combined dataset, it only performed best on accuracy. This suggests that training on data for the specific test set can make a big difference in certain evaluation metrics such as AUROC. Secondly, when training on CEDAR and testing on BHSig260, we see that DeepSign performs significantly better on all four metrics. This suggests we may have found a model that not only outperforms in evaluation metrics and has a smaller number of parameters, but may also be more generalizable to other languages or datasets overall.

Finally, we plot an ROC curve that demonstrates the tradeoff between sensitivity and specificty along a range of threshold values (Figure 2). Overall, we see that DeepSign overshadows the other two models.

Test Database	Model	Accuracy	Precision	Recall	AUROC
CEDAR Signature Database	SigNet	0.73	0.59	**0.65**	**0.79**
	MobileNetv2	0.72	**0.65**	0.28	0.72
	DeepSign	**0.74**	0.51	0.57	0.72
BHSig260 Signature Database	SigNet	0.81	0.69	0.75	0.85
	MobileNetv2	0.75	0.63	0.65	0.78
	DeepSign	**0.92**	**0.83**	**0.85**	**0.95**
Combined Database	SigNet	0.79	0.66	0.75	0.84
	MobileNetv2	0.76	0.65	0.70	0.82
	DeepSign	**0.85**	**0.76**	**0.84**	**0.93**

Table 6. Evaluation metrics for each database using models trained on the combined dataset.

Train/Test Database	Model	Accuracy	Precision	Recall	AUROC
CEDAR/CEDAR	SigNet	0.65	**0.70**	0.15	0.78
	MobileNetv2	0.68	0.67	0.22	0.72
	DeepSign	**0.76**	0.65	**0.59**	**0.80**
CEDAR/BHSig260	SigNet	0.65	0.53	0.26	0.64
	MobileNetv2	0.63	0.52	0.12	0.59
	DeepSign	**0.74**	**0.82**	**0.50**	**0.86**

Table 7. Generalizability of models to new languages. Evaluation metrics for models trained on one dataset and tested on the other.

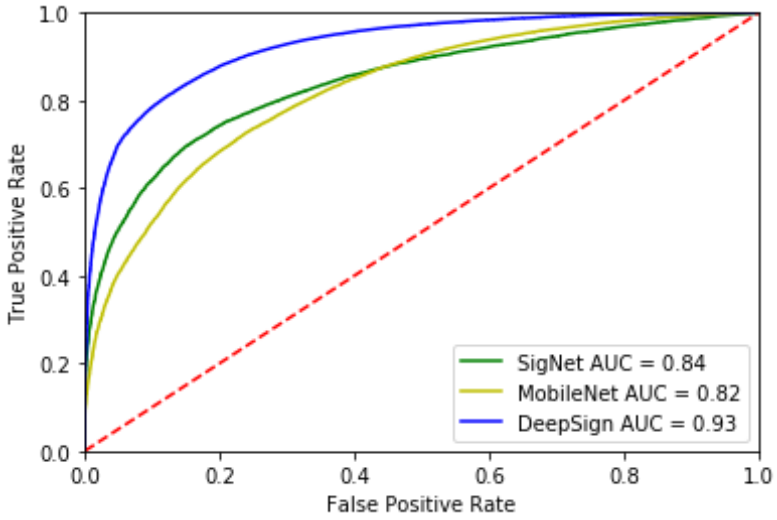

Figure 2. Receiver operating characteristic curve of SigNet baseline and SqueezeNet architectures demonstrates better performance of SqueezeNet by around 0.02

7. Qualitative visualization and evaluation

Recall that DeepSign consists of two identical networks which are fed $s1$ and $s2$ to generate two different embeddings. The distance between these embeddings is thresholded and used to predict whether the pairing is original-original or forged-original. We performed a qualitative visualization of these embedding distances on DeepSign's outputs by taking 9 image pairs that had high, medium, and low euclidean distances on a model trained on the combined data and tested on the combined data (Figure 3). Overall, we see that the algorithm does seem to learn some feature presentation related to the structure of the signatures, such that the euclidean distance between these embeddings is farther when signatures appear to be more different.

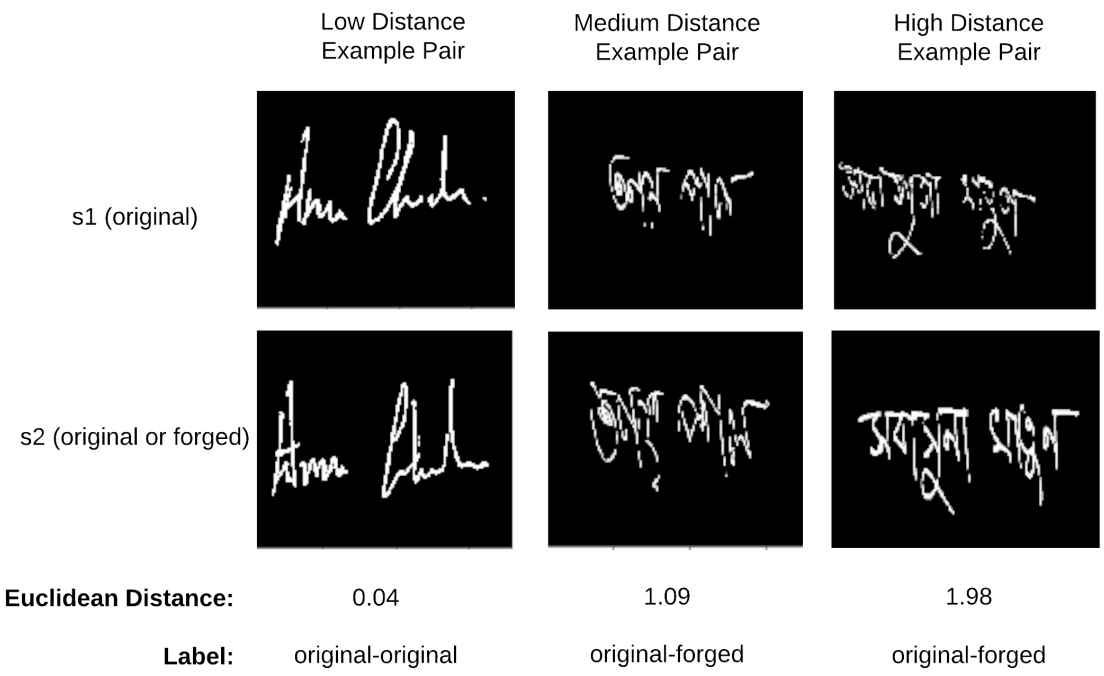

Figure 3. Three examples of high, medium, and low euclidean distance pairings from the training and testing on the combined dataset with DeepSign. Examples demonstrate how farther euclidean distances between embeddings also look less similar visually.

8. Conclusions and Future Work

We present a model, DeepSign, that has 25x fewer parameters and storage size and additionally outperforms SigNet, a state of the art signature verification model on both CEDAR and BHSig260 datasets in accuracy, precision, recall, and AUROC. Additionally, we show our model also outperforms a transfer-learned MobileNetv2, which is a state of the art mobile image classification model developed by Google. We also experiment with training on CEDAR, and testing on BHSig260. These results suggested that DeepSign may also be more robust for use on different languages and new data.

We suspect that DeepSign performed better than the MobileNetv2 as it was trained from scratch. Future work involves investigating training part of MobileNet or all of MobileNet and seeing how well it performs. We are also interested in performing a wider hyperparameter search, experimeting with SGD optimizers, and ensembling models to improve performance. Finally, we hope to develop an online web or mobile application for users to test the signature verification algorithm, as well as collect more training data (with user consent). The web app will be developed using React and TensorflowJS. We hope this web application will expand our dataset and draw more visibility to this field.

9. Contributions & Acknowledgements

Kevin Ko: Data pre-processing, deepsign, signet baseline

Jonathan Wang: Hyperparameter tuning framework, transfer learned model, literature review

We adapted code from the following sources for our project:
Contrastive loss function:
http://yann.lecun.com/exdb/publis/pdf/hadsell-chopra-lecun-06.pdf
Transfer learning for MobileNetv2:
https://keras.io/applications/

References

[1] Franck Leclerc and Rejean Plamondon. Automatic signature verification: The state of the art1989–1993. In *Progress in Automatic Signature Verification*, pages 3–20. World Scientific, 1994. 1

[2] Rejean Plamondon and Guy Lorette. Automatic signature verification and writer identificationthe state of the art. *Pattern recognition*, 22(2):107–131, 1989. 1

[3] Vishvjit S Nalwa. Automatic on-line signature verification. *Proceedings of the IEEE*, 85(2):215–239, 1997. 1

[4] Anil K Jain, Friederike D Griess, and Scott D Connell. On-line signature verification. *Pattern recognition*, 35(12):2963–2972, 2002. 1

[5] Dit-Yan Yeung, Hong Chang, Yimin Xiong, Susan George, Ramanujan Kashi, Takashi Matsumoto, and Gerhard Rigoll. Svc2004: First international signature verification competition. In *International conference on biometric authentication*, pages 16–22. Springer, 2004. 1

[6] Meenakshi K Kalera, Sargur Srihari, and Aihua Xu. Offline signature verification and identification using distance statistics. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(07):1339–1360, 2004. 1, 4

[7] Srikanta Pal, Alireza Alaei, Umapada Pal, and Michael Blumenstein. Performance of an off-line signature verification method based on texture features on a large indic-script signature dataset. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 72–77. IEEE, 2016. 1, 4

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[10] Sounak Dey, Anjan Dutta, J Ignacio Toledo, Suman K Ghosh, Josep Lladós, and Umapada Pal. Signet: Convolutional siamese network for writer independent offline signature verification. *arXiv preprint arXiv:1707.02131*, 2017. 1, 2, 3

[11] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994. 1, 2

[12] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 1

[13] Sheng Chen, Yang Liu, Xiang Gao, and Zhen Han. Mobilefacenets: Efficient cnns for accurate real-time face verification on mobile devices. In *Chinese Conference on Biometric Recognition*, pages 428–438. Springer, 2018. 2

[14] Klemen Grm, Vitomir Štruc, Anais Artiges, Matthieu Caron, and Hazım K Ekenel. Strengths and weaknesses of deep learning models for face recognition against image degradations. *Iet Biometrics*, 7(1):81–89, 2017. 2

[15] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 2, 3

[16] Hong-Wei Ng, Viet Dung Nguyen, Vassilios Vonikakis, and Stefan Winkler. Deep learning for emotion recognition on small datasets using transfer learning. In *Proceedings of the 2015 ACM on international conference on multimodal interaction*, pages 443–449. ACM, 2015. 2, 3

[17] Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, 2016. 2, 3

[18] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 2

[19] 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. 2

[20] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 2

[21] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. 2

[22] G. van Rossum. Python tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995. 2

[23] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006. 2

[24] Jeremy Howard. Cutting edge deep learning for coders. 3

[25] Min Lin, Qiang Chen, and Shuicheng Yan. Network In Network. *arXiv e-prints*, page arXiv:1312.4400, Dec 2013. 4

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5