Impact of Architecture Details on the Performance of Siamese Neural Networks

Vivek Cherangil Ramankutty
Matriculation Number: 1524431
Universität Siegen

Aswath Kolaron
Matriculation Number: 1524428
Universität Siegen

Abstract

Architectural details like the number of layers, neurons per layer, size of the output layer (i.e the final embedding), and the usage of regularizers like batch normalization and dropout layers influence the performance of Siamese Neural Networks (SNN). This was evident from the performance metrics of two different architectures namely SigNet[1] and SmallCNN[5]. So we analysed the impact of architectural details on the performance of SNN by performing a signature verification task using the BHSig260 dataset[3]. The study revealed that architecture with lesser trainable parameters and no regularizers like SmallCNN may perform well on some datasets but can get easily overfitted. But a heavier model like SigNet with regularizers can avoid overfit and achieve a global minimum in validation loss with fewer epochs.

1 Introduction

In tasks such as Face verification and Signature verification, identifying similarities between data instances is vital. The concept of similarity is subjective and hard to quantify and hence a typical machine learning algorithm like K-Nearest Neighbors or Support Vector Machine fails in these tasks. Whereas a Siamese Neural Networks (**SNN**) can understand ground truth based on the training pairs we provide. Its architecture consists of two identical networks which have the same structure and share the weights between them. As far as the neural network goes, it can be virtually be anything like a Convolutional Neural Network (**CNN**) or a Multilayer Perceptron (**MLP**). Instead of a single input instance, a pair of similar or dissimilar data instances are given as inputs. The network then learns to transforms these inputs to a new embedded space of lower dimensionality where the distance between them could be attributed to a direct measure of similarity.

By doing a literature survey on offline signature verification tasks, two different architectures namely SigNet[1] and SmallCNN[5] were found delivering 86.11% and 75.06% accuracy respectively on the same dataset. Even though, no research work was found explicitly analysing the impact of architecture details on the performance of Siamese Neural Networks. Hence we decided to study this open problem and understand how the number of layers, number of trainable parameters, and the usage of regularizers like batch normalization and dropout layers affect the performance of SNN.

2 Related Work

The convolutional SNN developed by Sounak et. al.[1], named SigNet, with four convolutional layers and two fully connected layers along with batch normalization, obtained an accuracy of 86.11% for Bengali signature and 84.64% for Hindi signature.

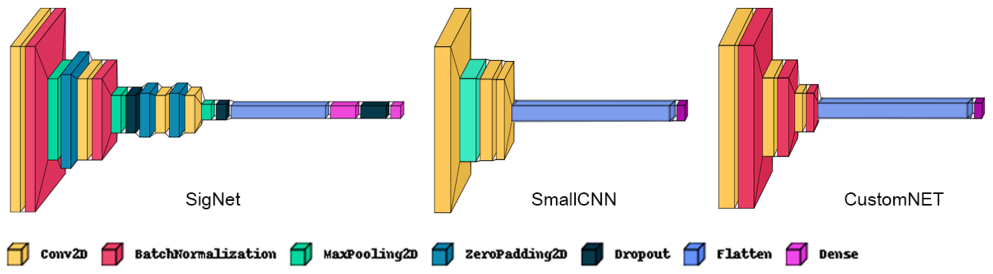
Rantzsch et.al.[4] used a network with a slightly reduced version of VGG-16. The work showed promising outcomes but the network has a large number of trainable weights which made the training process computationally demanding. The same goes with SigNet. Hence a very small CNN with only three convolutional layers and one fully connected layer, developed by Rateria et.al.[5] was also studied in this paper. The network had an accuracy of 75.06% for Bengali and 89.33% for Hindi signatures.

Batch normalization does help in reducing the time involved in training and minimizing the internal covariate shift occurring when the model is subjected to a new set of data which in turn causing the hidden activation to shift[2].

3 Methodology

For the analysis, the networks chosen were, **SigNet** [1], **SmallCNN** [5] and **CustomNet** (Self-made model). The detailed architectures of these networks can be found in Tables 2, 3 and 4 respectively.

Figure 1: Network Structures



3.1 Network Architecture Comparison

All these networks start with a CNN as it can extract the features far better from an image than an MLP. In general, these architectures vary in the number of layers and neurons per layer, the size of the output layer (i.e the final embedding), and in the usage of regularizers like batch normalization and dropout layers. Pooling layers are also used throughout the network to sample down the feature map. Which in turn reduces the number of model parameters.

The SigNet architecture has four convolutional layers and two fully connected layers. Batch normalization and dropout layers were also added to enhance the generalization capability. The downsampling at the initial convolutional layer is also substantially higher in SigNet. SmallCNN has only three convolutional layers and one fully-connected layer with no regularizers. Overall SigNet is having 6.46 Million parameters, whereas in SmallCNN it is only 2.97 Million. CustomNet was developed to take a middle ground with 5.82 Million parameters and regularizers similar to that of SigNet. It has three convolutional layers and two fully connected layers.

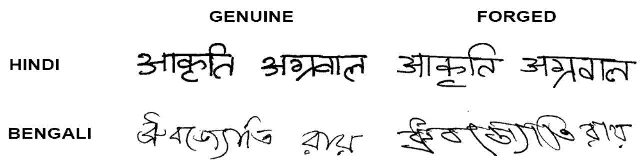
3.2 Data Pairing & Split-up

The dataset used is BHSig260 introduced by Pal et. al. [3] containing the Bengali and Hindi signatures. There are 160 classes in the Hindi signature set and 100 in the Bengali. Each class consists of 24 genuine and 30 forged signatures from which the pairs have to be generated. The Genuine-Genuine pair will be having the label 1 and for Genuine-Forged, the label would be 0. The 160 classes in the Hindi dataset are split for training, testing, and validation in the ratio of 120:20:20. In the case of the Bengali dataset, the total 100 classes are split in the ratio of 70:15:15 (Train:Test:Validation).

In each class, there will be ${}^{24}C_2 = 276$ Genuine-Genuine image pairs and to create the Genuine-Forged pairs, all 24 of the genuine signatures of a class is paired with 12 randomly selected forged signatures from the total of 30 of the same class. So, the total number of pairs of images that can be generated for the Hindi set will be 67,680 for training and 11,280 each for testing and validation. For the Bengali set of signatures, there will be 39,480 image pairs for training and 8,460 image pairs each for testing and validation. The batch size is set to 128 giving a total of 528 batches for the Hindi dataset and 308 batches for the Bengali. The pairs of images along with the label are shuffled before passing into the model.

The sample of genuine and forged images from both datasets are shown in figure 2.

Figure 2: Sample signatures



3.3 Preprocessing

Architectures taken from literature for this study have shown their best results with different preprocessing methods. So to achieve generalization, images were resized to a dimension of 155 x 220 and normalized before feeding into the model.

3.4 Model and Algorithm Hyperparameters

Rectified Linear Units (ReLU) is the activation function used in all the models, and the weights were initialized with the 'glorot_uniform' method in Keras. Models like SigNet and CustomNet were compiled using RMSprop optimizer and ADAM was chosen for SmallCNN. The contrastive loss is used with all the models. Even though, initial learning rate (LR) was set to $1e^{-4}$, ρ = 0.9 and ϵ = $1e^{-8}$ for the RMSprop, it was updated using "ReduceLROnPlateau" via Keras callbacks if the validation score doesn't improve over 5 epochs. For ADAM, the learning rate and epsilon values are taken similar to that of RMSprop. Most hyper-parameters were directly taken from the literature and training was done on both Hindi and Bengali datasets. Early stopping was also applied to stop training if the validation loss doesn't improve for 10 epochs.

3.5 Machine

We used google colaboratory for running the code implemented in Keras with TensorFlow backend and it is made available on GitHub. [1] The GPU allocated by colab was Tesla T4 having 2560 CUDA cores.

3.6 Performance Metrics

From the literature, we considered five commonly used performance metrics and are Accuracy, True Positive Rate (TPR), True Negative Rate (TNR), False Rejection Rate (FRR), and False Acceptance Rate (FAR).

To calculate the accuracy, a threshold has to be set to differentiate the genuine from the forged. Hence a well-known approach called Receiver Operator Characteristics (ROC) Curve was used. In this approach after the distances between the image pairs in the test data are predicted, the threshold value is varied between the least and highest distance. For each threshold value, the accuracy is calculated and the one which gives the highest accuracy is selected.

In tasks like signature verification, a FAR value which is a measure of the likeliness of forged signatures being accepted is also critical.

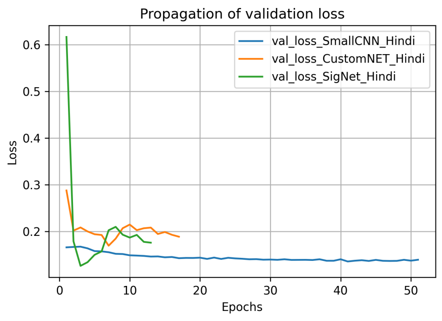
4 Results

4.1 Performance Stats

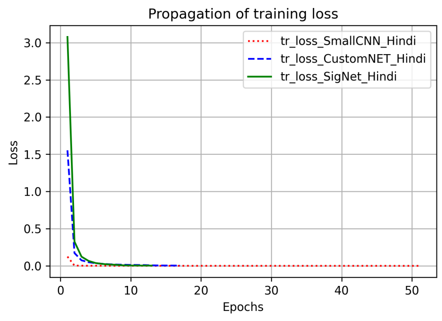
The propagation of validation and training losses for both Hindi and Bengali sets of data in all the models are compared in the following plots 3 and 4.

From the plots, we observed that the difference in the magnitude of validation and training losses at the beginning depends on the number of model parameters and the presence of regularizers. Even

[1] `https://github.com/vvekraman/Architecture-Analysis-Siamese-Neural-Networks`

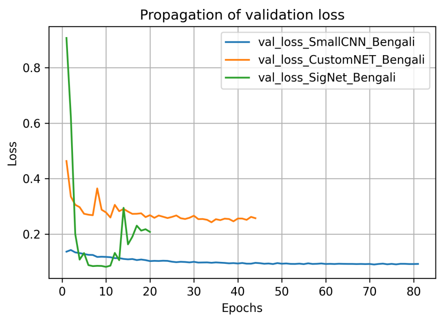


(a) Validation loss

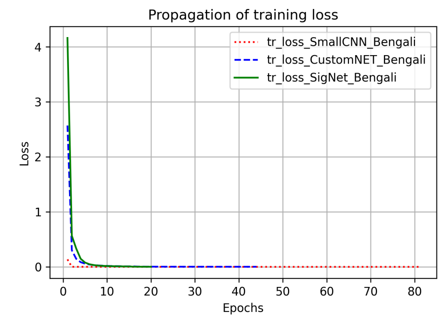


(b) Training loss

Figure 3: Trend in Validation loss & Training loss - Hindi dataset



(a) Validation loss



(b) Training loss

Figure 4: Trend in Validation loss & Training loss - Bengali dataset

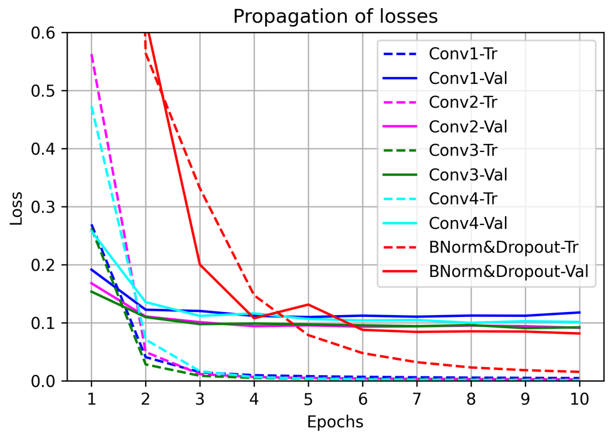
though there is a significant difference in the losses initially, the SigNet is capable to attain a global minimum with fewer epochs. Also, while comparing the training losses at the best validation score, the SmallCNN is in overfit category compared to the rest. The performance metrics of SigNet and SmallCNN are almost equal but the time to best validation score and the training loss lying significantly close to zero does make the SmallCNN less preferable. The CustomNET has somewhat comparable performance stats in the Hindi dataset but, it performed worse with the Bengali set of signatures. The values of the performance metrics of all the models are given in Table 1.

Table 1: Performance parameters

	SigNet		SmallCNN		CustomNet	
Parameters	Hindi	Bengali	Hindi	Bengali	Hindi	Bengali
Accuracy (%)	**83.98**	91.08	82.17	**91.25**	80.92	75.54
TPR (%)	**88.65**	93.52	85.52	**95.43**	79.18	73.12
TNR (%)	79.30	**88.63**	78.82	87.08	**82.65**	77.97
FAR (%)	20.69	**11.36**	21.17	12.91	**17.34**	22.02
FRR (%)	**11.34**	6.47	14.47	**4.56**	20.81	26.87
Threshold	0.53	0.64	0.54	0.55	0.49	0.41
Time/epoch (mins)	3.5 to 4	2.5 to 3	4 to 4.3	2.5 to 2.8	$\sim$4	2.8 to 3
Best validation loss	**0.1264**	**0.0815**	0.1359	0.0900	0.16973	0.2424
Training loss	0.1234	0.0154	3.2089e-04	1.1612e-08	0.0212	0.0021
Epochs to best validation loss	**3**	**10**	41	71	7	34

To understand the influence of each layer on the losses, the SigNet model is stripped down to the first convolutional layer, pooling layer, padding layer, flatten layer and two dense layers and is run for 10 epochs to collect the loss stats. Now each convolution layer along with the pooling and padding layer is added one after the other and the propagation of losses are analysed. With the addition of batch normalization and dropout layers, the final SigNet model is obtained.

Figure 5: Loss plot - SigNet analysis



From the plot 5, with only the first convolutional layer, we observed a significant gap between the validation and training losses. This puts the model in overfit category. With the addition of the second and third convolution layers, a slight improvement in the validation loss is seen, but the model was still getting overfitted easily. The validation score got poor by the addition of the fourth convolution layer. By the addition of the batch normalization and dropout layers, we can see a significant improvement in the loss propagation. The validation loss achieved much better values in fewer epochs without the model getting overfitted.

5 Conclusion and Future Works

Even though the performance metrics of SmallCNN and SigNet were comparable, the SmallCNN model was overfitted at its best validation score. From our experience, an overfit model may tend to perform worse when subjected to a new type of data. The performance of the models also depends a lot on the dataset used, as there isn't a generalised model which performed well with all types of data.

The heavier model, SigNet when analysed, we understood the significance of the usage of regularizers like batch normalization, dropout, and weight initializers which did help in avoiding overfit and achieving a global minimum in validation loss with minimum epochs.

As the decentralization of data processing is the key aspect in Edge Computing, the lighter models are a major requirement when it comes to deployment in low-power devices. In our research, pruning could be a way to reduce the network size, or we could also use models like SmallCNN with regularizers to avoid overfitting.

In our case, the computational resources available were limited. If resources were not a bottleneck, Grid Search or K-Fold Cross-validation could be used to tune both model and algorithm hyperparameters. The influence of each layer of the network on the embedding capability could also be further analysed . During preprocessing, more network-specific transformations could be applied in the future to extract more details from the images and to improve diversity among samples.

References

[1] 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.

[2] Jamie Dowat. Internal Covariate Shift: How Batch Normalization can speed up Neural Network Training. `https://bit.ly/3r29M6U`. Accessed: 2021-07-11.

[3] 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, 2016.

[4] Hannes Rantzsch, Haojin Yang, and Christoph Meinel. Signature embedding: Writer independent offline signature verification with deep metric learning. In *International symposium on visual computing*, pages 616–625. Springer, 2016.

[5] Avani Rateria and Suneeta Agarwal. Off-line signature verification through machine learning. In *2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, pages 1–7. IEEE, 2018.

A Appendix

Table 2: Network Architecture - SigNet

Layers	Filters	Kernel size	Strides	Output size
Input	-	-	-	155x220
Convolution	96	11x11	4	37x53
Batch Normalization	96	-	-	37x53
Max Pooling	96	3x3	2	18x26
Zero Padding (2,2)	96	-	-	22x30
Convolution	256	5x5	1	18x26
Batch Normalization	256	-	-	18x26
Max Pooling	256	3x3	2	8x12
Dropout	256	-	-	8x12
Zero Padding (1,1)	256	-	-	10x14
Convolution	384	3x3	1	8x12
Zero Padding (1,1)	384	-	-	10x14
Convolution	256	3x3	1	8x12
Max Pooling	256	3x3	2	3x5
Dropout	256	-	-	3x5
Flatten	-	-	-	3840
Dense	-	-	-	1024
Dropout	-	-	-	1024
Dense	-	-	-	128

Table 3: Network Architecture - SmallCNN

Layers	Filters	Kernel size	Strides	Output size
Input	-	-	-	155x220
Convolution	40	5x5	1	151x216
Max Pooling	40	2x2	2	75x108
Convolution	30	3x3	1	73x106
Convolution	20	3x3	1	71x104
Flatten	-	-	-	147680
Dense	-	-	-	20

Table 4: Network Architecture - CustomNet

Layers	Filters	Kernel size	Strides	Output size
Input	-	-	-	155x220
Convolution	80	11x11	2	73x105
Batch Normalization	80	-	-	73x105
Max Pooling	80	2x2	2	36x52
Convolution	40	5x5	1	32x48
Batch Normalization	40	-	-	32x48
Max Pooling	40	3x3	2	15x23
Convolution	20	3x3	1	13x21
Flatten	-	-	-	5460
Dense	-	-	-	1024
Dropout	-	-	-	1024
Dense	-	-	-	128