Comparison on Performance Between Different Models on Identification of Sparse Natural Person Photos

Meilin Guo

Research School of Computer Science
The Australian National University
Email: u7094624@anu.edu.au

Abstract. This paper is about identification for matching to sparse natural person photos using three kinds of models including simple feed-forward neural network [4], bidirectional neural network [1] and convolutional neural network [3]. It is aiming to identify whether the two images are of the same person given proportions for distances of facial feature marker pairs [2]. I compare the best performances in 20 times for each model with identical settings since the way I partition the unbalanced sparse dataset may cause unstable results. All three of my models lead a superior performance than that in the dataset paper [2], with the order of superior: Feed-forward NN<BDNN<CNN models.

Keywords: Bidirectional neural network, Unbalanced sparse data, Convolutional neural network, Supervised machine learning, Feed-forward neural network

Section I Introduction

Photographs play a significant role in preserving memories and knowledge in a historical way. When a natural disaster strikes, the first thing survivors often mourn is a lost photograph. [1] However, countless historical photographs, from decades of their origins, are orphaned, face images without names, and no living person is able to identify the individual in the photograph even if the photograph is in the public domain, such as the archives of the Australian War Memorial and the National Library of Australia.[1]

The simplest multi-layered Feed-forward neural network [4] is proposed by Svozil to increase the capability of generating more information. It proposed to have multiple layers instead of single to investigate deeper. As the advance of technology, bidirectional neural networks and class prototypes are proposed by Nejad, Akbar and Tamás in 1995 which is based on multi-layer perceptrons trained by a generalized form of the error back-propagation algorithm [4]. I think these two neural network models are able to perform binary classification on photos and identify match or not a match. Therefore, I'm inspired to build these two kinds of models. The convolutional neural networks give well performance on image processing since it generates 2D inputs and down-sample to preserve the most significant information and end up with fully connected layers. I think this network may have better results in identification on person photos as well, hence I decided to perform on this model in addition.

In this paper, I sought to investigate more on comparisons on performances between three kinds of neural networks, which are Feed-forward neural network (Feed-forward NN), bidirectional neural network (BDNN) and convolutional neural network (CNN).

Section II Method

A. Dataset

a. Dataset Inspection

There are two kinds of datasets are provided along with description corresponding to the paper by Dr Sabrina Caldwell [2]. These are of a specific category of images: two-dimensional still camera photographs of natural persons. Furthermore, we focus more on the case where only sparse photos are contained here. We have 12 sets of 3 images, where Image A & B are identified matching with the same person but Image C is identified not matching with neither

Image A nor Image B. Hence, we have 12*3=36 items here where the first item outputs 1 (denotes matching) following two items outputting 0 (denotes not matching). [11]

I chose the dataset formulated by proportions for distances between 14 facial feature markers into 91 new features, as shown in Fig. 1 and Fig. 2, instead of choosing the dataset containing only coordinates of 14 facial feature markers because no two photographs to be compared will be the exact same size and aspect. Therefore, it isn't enough to simply know the distance between points in the image to have useful information to compare against a different image of potentially the same individual. We need to work out the relationships of these distances in comparison to one another in order to compare two different photographs.

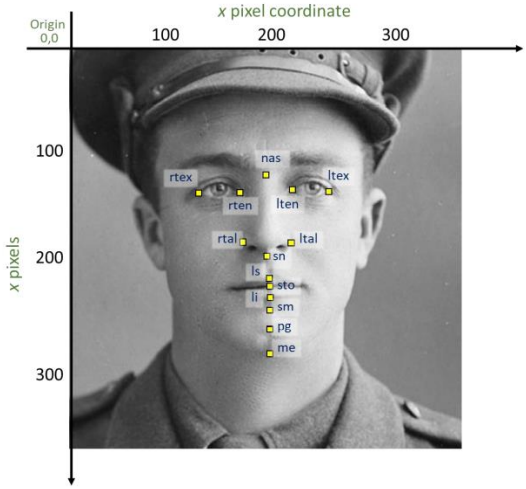

Fig. 1. Set of 14 facial markers depicted [2]

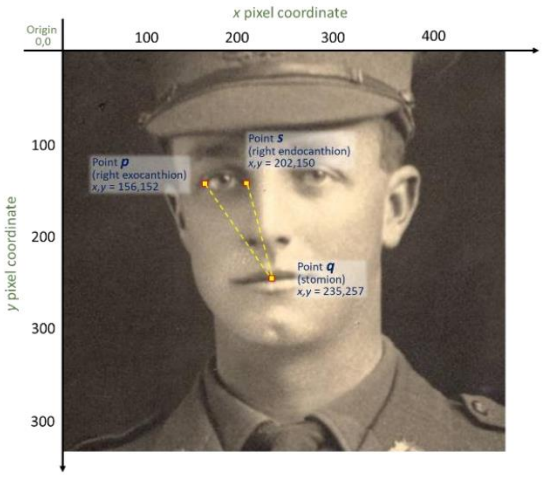

Fig. 2. Calculating distance between facial markers in an image [2]

b. Dataset Exploration

The type of datasets is basically the float type of all values. It is in the form of excel in sheet 3 with 36 rows and 183 columns (apart from the index column and the feature names' row) including 91 proportions for distances between 14 facial feature marker pairs diploid due to two images' information in one line with the identification of matching or not a match denoting the two classes. It doesn't need normalization or addition, mergence and pruning on variables since we'll just feed each data with the input size of 182 into feed-forward and bidirectional neural network models. As for the convolutional neural network, we generate 182 into 13*14 for each data item to gain 2D inputs, hence no adjustment about the data features requires at all in this way.

However, an adjustment which is performed corresponding to bidirectional neural network paper is adding an extra node (i.e. an extra feature) [1], which aims to make the associative memories invertible (the same idea of one-to-one function in [1][8]) and increase the generalization and the reliability of neural networks. An classical problem namely XOR problem and the way using an output extra node is shown in Table 1, which has achieved in forming a one-to-one mapping. The reason I abandoned that step is that I successfully realized the reverse process manually by writing a reverse function within BDNN model by multiplying the weights and values inversely. This is realizable mainly due to the output layer has Softmax activation function, instead of binary outputs.

Table 1. XOR problem and using output extra node[1]

Inputs			Output	Extra node
0	0	0	0	0.1
0	1	1	0	0.3
1	0	1	0	0.5
1	1	0	0	0.7
0	0	1	1	0.2
0	1	0	1	0.4
1	0	0	1	0.6
1	1	1	1	0.8

c. Dataset Preparation and Findings

The obvious problem we need to tackle first with the dataset is that the dataset contained in sheet3 which are proportions for distances between facial feature marker pairs are extremely unbalanced and sparse for positive class, which is one third of the size of dataset. There exist several methods to ease the limitation and extend dataset. Ruth G. Shaw and Thomas Mitchell-Olds have proposed the ANOVA to analyzing data using analysis of variance [10]. The easiest way to successfully generalize a model is to use more data. The problem is that out-of-the-box classifiers like logistic regression or random forests tend to generalize the model by dropping rare classes. When data conform to a complete, balanced design (equal numbers of observations in each experimental treatment), it is straightforward to conduct an ANOVA, particularly with the aid of the numerous statistical computing packages that are available. However, it is apparent that it's not suitable for classify the binary outputs (positive for 1 and negative for 0) here.

Table 2. Partition on training and test sets

	Positive class	Negative Class	Sum
Train Set	8	17	25
Train Set	4	7	11
Sum	12	24	36

The technique I used here for the dataset is manually split the data into train data and test data, as shown above in Table 2. Since we only have 36 items w.r.t. 2 classes (positive and negative for matching),we can manually split data to ensure the balance. I try several patterns to evaluate for the optimal, then I finalize to randomly split data into 25 out of 36 for training sets and the remaining 11 for test sets. This also makes the datasets disjoint to each other. To ensure the balance and the credibility, I manually set random 4 positive class and 7 negative class for the test set. I shuffle the dataset manually, then select the first 4 positive data and 7 negative data into test, which is capable of achieve random. Note that this is inspired by the method implemented in the dataset paper [2]. However, I made a change on the ratio of test/train sets for 11/25 instead of 12/24, which is mainly for the size of training data to have an integer multiple of the batch size to ensure the capacity of training reversely in the directional neural network model.

I use this method for all models I have implemented. The most pitfall with this method is that it also leads the results (such as accuracy) having a large variance related to diverse partitioning on dataset. Therefore, we need to alleviate this problem by means.

B. Architecture of Models

There are three kinds of models I have applied to investigate performances on, which are feed-forward neural network, bidirectional neural network and convolutional neural network. The basic concepts are discussed as below.

a. Feed-forward Neural Network

A classical Feed-forward neural network is demonstrated below in Fig.3. The model I applied is much similar to this sample. The Feed-forward NN gets the fixed size of inputs then fully connected to the first hidden layer, so as the connection to the following hidden layers, ultimately an output layer. Weights are the values on relationship between neurons in connected layers while biases are added to neurons. Activation functions are determined to sensitize the layer forward. We could set the estimated cell to Softmax for classification or other approaches like Log and Tanh for regression. The weights are updated corresponding to optimizer by back propagation.

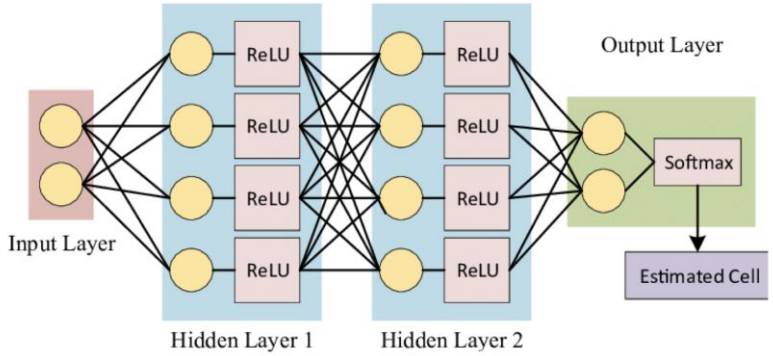

Fig. 3. Sample Topology of Feed-forward NN [7]

b. Bidirectional Neural Network

The idea of bidirectional neural network is slightly superior to the feed-forward neural network, as a result of it takes into account that the output is related not only to previous sequence elements, but also to subsequent sequence elements.

This model has a similar architecture as the feed-forward neural network, but it can be running on both directions: from inputs to outputs and from outputs to inputs, with shared weights and separate bias on the forward and reverse processes, as s shown on Fig.4. As we know that, backpropagation is applied to update and optimize weights from output layers, each direction of process on BDNN can have a impact on weights on another direction since they are shared, in turn cause effect on loss and accuracy.

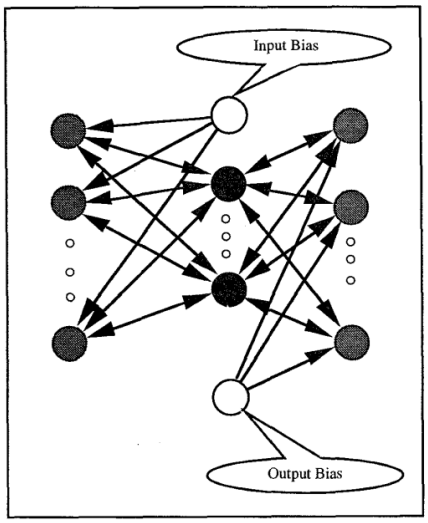

Fig. 4. BDNN Topology[1]

c. Convolutional Neural Network

We can consider a convolutional neural network is more generate since it can have a more comprehensive aspects on inputs, comprising data and the position (not only the front and back sequences like BDNN), because the CNN input is 2D instead of 1D with only a sequence [3]. The classic and most popular example of using CNN networks is image processing.

The sample topology of CNN is shown in Fig. 5. In the sample topology, the input is of shape 23*28, feeding into the first convolution layer to have 20 channels. Then it down-samples into one forth size, and feeds into convolution layer and down-samples again. Generally speaking, CNN model will end up with a fully connected layer with specified output size.

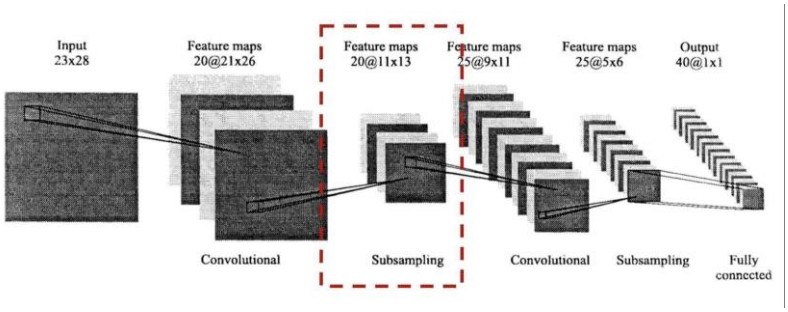

Fig. 5. Sample CNN Topology [5]

C. Evaluation

As for the implementation on both training and testing the bidirectional neural network, the model is designed to process forward then backward with one batch. The loss calculated in bidirectional neural network is the sum of both ways.

I have defined variables to record history with the accuracy and loss change for training and testing for each model. An supervision is conducted to ensure a better performance on models.

Comparison on Performance Between Different Models on Identification of Sparse Natural Person Photos

I evaluate the performance of each model by the best performance during 20 times of tryout, which is due to the weak robustness of data partitioning on training and testing datasets. Then I fine-tune the hyperparameters to achieve better results. Note that this is not a best way of evaluation. A better method is considered in the future work.

Section III Implementation

a. Settings and Parameters

After varieties of attempts on parameters, like loss and accuracy change on both training and testing datasets, we finalize to have the parameters shown in Table 3.

Table 3. Parameters of three models

	Feed-forward NN	**BDNN**	**CNN**
Number of Epoches	50	50	50
Learning Rate	0.02	0.02	0.02
Loss Function	Cross Entropy Loss	Forward: Cross Entropy Loss Backward: MSE	Cross Entropy Loss
Optimizer	Adam	Adam	Adam
Batch Size	5	5	5
Topology	Input Layer :1*182		Input size: 13*14
	Hidden Layer 1: 30, ReLU		Conv2d:1,5,3*3, ReLU
	Hidden Layer 2: 10, ReLU		Conv2d:5,10,5*5, ReLU
	Output Layer: 5		Maxpooling:2*2,padding=1
			Fully Connected: Input:40, output:5, Softmax

b. Pipeline

1. Datasets

I load the data from *sheet2/facial-features* excel file to variable *data* and drop the first column as it is the identifier. I shuffled the data manually using *sample* function which is built in *pandas* package. I used variables *data_match* and *data_not_match* to denote positive and negative classes by the value of *output* column. Then I pre-define some hyperparameters for splitting datasets. It is sufficient for us to have two hyperparameters located the size and distribution of positive and negative data points in training and test sets. The two hyperparameters I define here is the

number of positive data in test set and the size of test set, denoted by *test_match_num* and *test_num.* As a result, *test_match, test_not_match, train_match* and *train_not_match* are calculated by fraction on *sample* function which is all computer by the two hyperparameters. Therefore, it ensures the feasibility of codes. The positive and negative classes are concatenated separately to formulate training and test datasets, denoted by *train* and *test.* The number of features is represented by *n_features,* which is simply the result of number of columns in train set minoring one. We also split features and output for convenience to calculating loss. Ultimately, train and test tensors are defined to hold inputs and outputs. I load the data to fit into models.

2. Training and Testing

Hyperparameters for each model are defined in the first place, which are *batch_size, learning_rate* ad *epochs.* History records are defined before training and testing for plots, which are *train_loss, test_loss, train_acc* and *test_acc.* The model's mode is set to *train()* in advance. The accuracy and loss are reset to zero before each batch. During each batch, I set the optimizer to have zero gradients. Loss is calculated by inbuilt function in *torch.nn.function* package called *cross_entropy* by feeding the labels and outputs from models. Note that loss on BDNN model is calculated by the sum of forward and backward. Eventually, we activate loss to back propagate and optimize weights.

The testing process is merely identical to training but obtaining data from *test_loader* and set the model's mode in *eval()* in advance instead.

Section IV Results and Discussions

a. Results

The loss and accuracy change on the best settings for three kinds of models over epochs is shown in Fig.6-11.

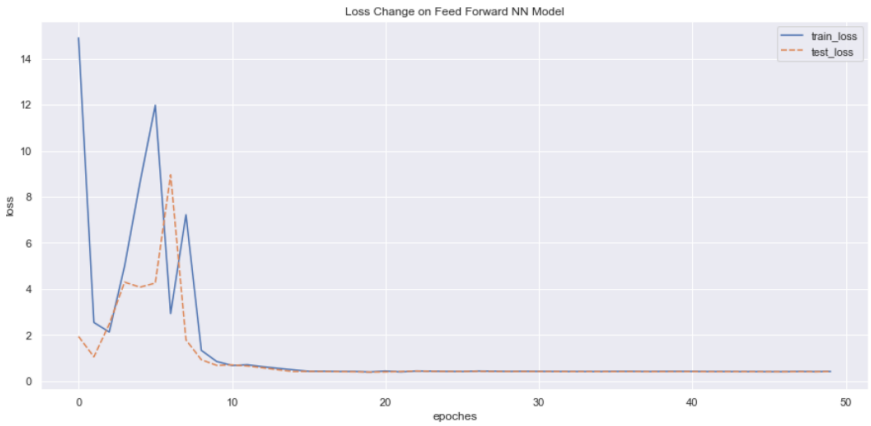

Fig. 6. Loss Change on Feed-Forward NN Model

Comparison on Performance Between Different Models on Identification of Sparse Natural Person Photos

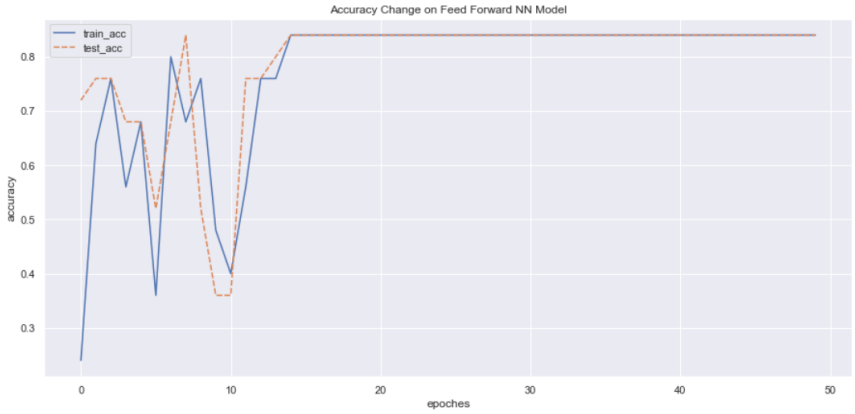

Fig.7. Accuracy Change on Feed-Forward NN Model

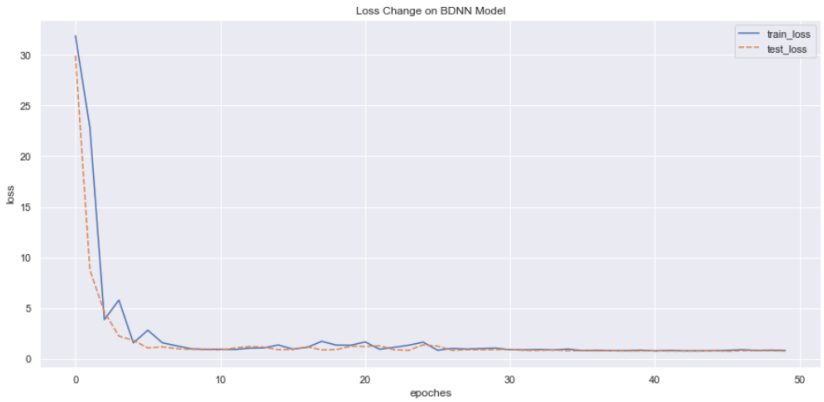

Fig. 8. Loss Change on BDNN Model

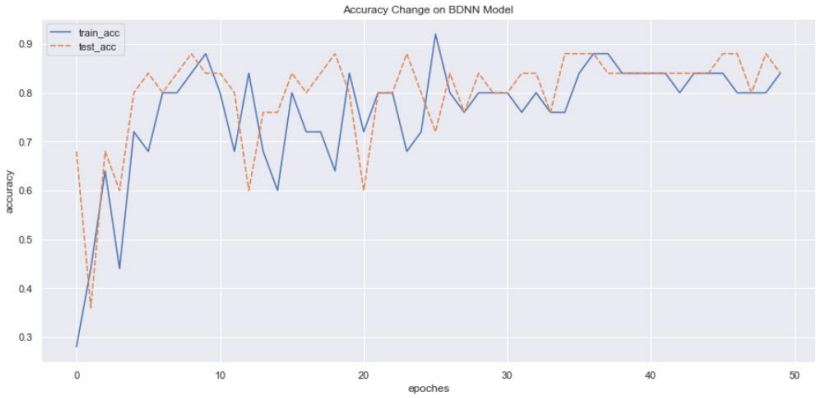

Fig. 9. Accuracy Change on BDNN Model

Comparison on Performance Between Different Models on Identification of Sparse Natural Person Photos

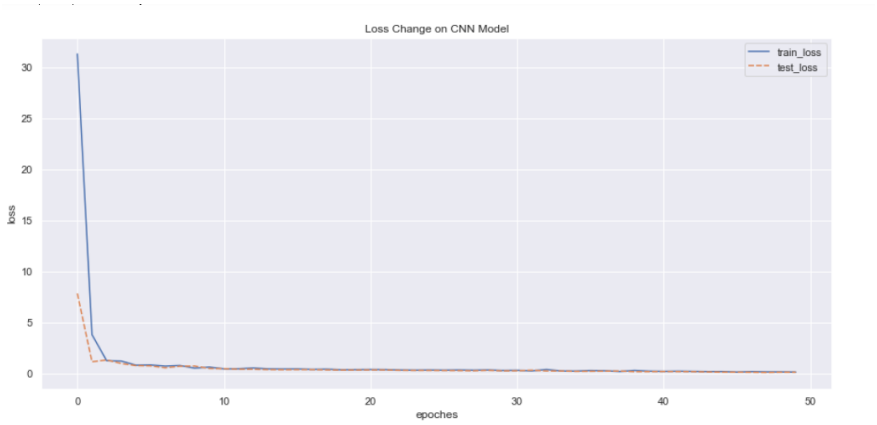

Fig. 10. Loss Change on CNN Model

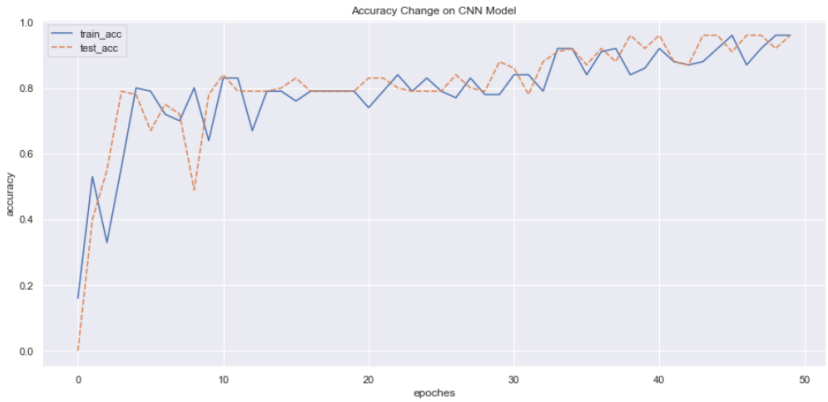

Fig. 11. Accuracy Change on CNN Model

b. Discussions

1. Performance Comparison on Dataset Paper

The classification neural network in dataset paper [2] contained 2 hidden layers, with 10-fold cross validation to mitigate the effect of the small dataset size. The training was set to 24 out of 36 feature sets and testing was set to the remaining 11 out of 36 feature sets. The test set was set to 4 'match' and 8 'not a match' [2]. The best accuracy that model could achieve was 75%, which was the average across 10 iterations of 10-fold classification [2].

As for my accuracy on models, the accuracies on all three models are greater than 80%, with a maximum value of 96% and the minimum of 80%, which suggests superior capability of identification of sparse natural person photos.

2. Performance Comparison Between Three Models

Convergence time consumption:
As the results shown above, the Feed-forward NN model finalizes to converge at epoch 13 in a faster way, while the BDNN tended to be stable at epoch 35, with a longer time consumption to converge than the Feed-forward NN model

since the update on weights with be affected by the reverse direction process on BDNN model, which gives a great possibility to change the optimal weights. Therefore, it requires longer time. However, the result shows within a normal range. The slowest one is CNN model, which is since it may need more time to get information in 2D and have more layers, not only simply fully connected layers.

Accuracy and Loss:

The CNN model performs the best on accuracy of classification on outputs, with a maximum value over 95% and a nearly zero loss. This model is able to obtain more information on natural person photos. Therefore, it is unsurprised to have the best accuracy.

The accuracy on the BDNN model is the worst among these models, with a minimum value of just over 80%. In the meanwhile, we can see from the accuracy change on BDNN model during the middle part from epoch 10 to epoch 28 that, the accuracy tends to fluctuated a lot while the loss doesn't change much. I think the reason behind this phenomenon remains the instability caused by shared weights. The loss and accuracy are computed after each batch of back and forth, while the better results may be affected by the reverse direction of process, which may lead to the efforts did before in vain.

The loss for all three models is of a approximate zero value, while Feed-forward NN model may lead a highest loss, but still, lower than 0.5. The accuracy is between BDNN model and CNN model, since it can only obtain 1D information but is free of interruptions backward.

Section V Conclusion and Future Work

The most drawback of my model is lack of validation system to fine-tune a better hyperparameter and evaluate models, like K-fold Cross Validation [9]. It will be better to evaluate models just based on accuracy of test sets by multiple attempts. I'll improve that in my future work. In the meanwhile, as for the optimizer, I have applied Adam for all of three neural network models, while an optimizer, namely AdamW, is well-known to converge faster than other common optimizers like SGD (stochastic gradient descent) [6]. It may lead to better results on a shorter time. Therefore, I'll also tryout this in the future.

As for the implementation procedure on BDNN, I have successfully implemented by using the back-and-forth method, which means a batch of data is training and testing for both forward and reverse at one epoch. Meanwhile, the loss is calculated by the sum of both ways. Another procedure I'm able to do in the future is implementing the alternative method, which is of the process that the model is fully trained on one side with a threshold on loss calculated that way, which will stop with a lower loss by threshold, then switch the direction. It can be designed to stop either by the number of epochs or the threshold on loss like the first half process. Note that this is inspired by Hossain and Gedeon's paper [9].

In conclusion, all three of my models lead a superior performance than that in the dataset paper [2], with all accuracies greater than 80%. Moreover, CNN fits the best to classification on each pair of photos among them, while BDNN finalizes the worst due to consumption backward process caused. The Feed-forward NN model expresses the average performance.

References

[1] Nejad, Akbar Farhoodi, and Tamás D. Gedeon. "Bidirectional neural networks and class prototypes." Proceedings of ICNN'95-International Conference on Neural Networks". Vol. 3. IEEE, 1995.

[2] Caldwell, S. (2021) "Human interpretability of AI-mediated comparisons of sparse natural person photos," CSTR-2021-1, School of Computing Technical Report, Australian National University.

[3] Shin, Hoo-Chang, et al. "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning." IEEE transactions on medical imaging 35.5 (2016): 1285-1298.

[4] Svozil, Daniel, Vladimir Kvasnicka, and Jiri Pospichal. "Introduction to multi-layer feed-forward neural networks." Chemometrics and intelligent laboratory systems 39.1 (1997): 43-62.

[5] Tom, S1,2021, Lecture notes, DL3-Convolutional Neural Networks, page 38, COMP 4660/8420, Bio-inspired Computing: Application and Interfaces, The Austrian National University, adapted from lecture notes by Christopher Chow ad Josephine Plested.

[6] Loshchilov, Ilya, and Frank Hutter. "Fixing weight decay regularization in adam." (2018).

[7] Belmonte-Hernández, A., Hernández-Peñaloza, G., Gutiérrez, D.M. and Alvarez, F., 2019. SWiBluX: Multi-sensor deep learning fingerprint for precise real-time indoor tracking. IEEE Sensors Journal, 19(9), pp.3473-3486.

[8] Jacobsen, Jörn-Henrik, Arnold Smeulders, and Edouard Oyallon. "i-revnet: Deep invertible networks." arXiv preprint arXiv:1802.07088 (2018).

[9] Hossain, M.Z. and Gedeon, T., 2017, May. Classifying posed and real smiles from observers' peripheral physiology. In Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare (pp. 460-463).

[10] Shaw, Ruth G., and Thomas Mitchell-Olds. "ANOVA for unbalanced data: an overview." Ecology 74.6 (1993): 1638-1645.

[11] Meilin Guo, " Bidirectional Neural Network on Identification of Sparse Natural Person Photos" , the 4th ANU Bioinspired Computing Conference.