LVA-Number: 344.031

Practical Work in AI

LVA-Leader : Oleg Lesota / Markus Schedl

# Face Recognition with Neural Networks

Author:

Boran Cihan Polat

K12055254

Submission date: 08. Februar. 2024

# Table of Contents

# 1. Introduction

## 1.1 Motivation

In today's world, AI is everywhere in our daily lives. One of in my opinion the most interesting uses is recognizing faces. We use it on our phones, tablet, pc, etc., to identify ourselves with the technology. The motivation for this project stems from the increasing importance of face recognition (FR) technologies in various applications, from security systems to personal device authentication. The exploration of Convolutional Neural Networks (CNN's) and their variants has shown promising advancements in FR. However, the challenge remains in enhancing the accuracy and efficiency of these models through innovative approaches.

## 1.2 Research Questions

The primary research question seeks to identify the most optimal and interesting approach for FR. After reviewing various FR-Methods, the project introduces specific questions of the Siamese Neural Network (SNN) (Koch, Zemel, & Salakhutdinov, 2015) and its distinct characteristics. Furthermore, an investigation into how various loss functions and their hyperparameters influence the enhancement of network accuracy and efficiency. Additionally, we examine the impact of using ReLU versus SeLU (Guo & Zhang, 2019)activation functions. These focused questions aim to deepen our understanding of advanced neural network applications in FR, highlighting the significance of network architecture and loss optimization in achieving superior recognition outcomes.

**Overview of the research questions:**

1. Which Neural Network architecture demonstrates the most effective performance for FR tasks?
2. What are the unique strengths of SNN?
3. Which loss functions optimize the performance of the chosen network architecture for FR application?
4. Which Activation function optimizes the performance of the chosen network architecture for FR application?
5. How do different hyperparameters settings within the chosen loss function impact the network's accuracy and efficiency?

## 1.3 Outline

The report is structured as follows: After the introduction, Section 2 covers the materials and methods, detailing the data and methodologies used. Section 3 describes the implementation process of the Siamese neural network. Section 4 presents experimental results, examining the impact of various loss functions on network performance. Finally, the conclusion and outlook provide a summary of findings and potential future directions for research in this field.

# 2 Material and Methods

## 2.1 Dataset

We used the Labelled Faces in the Wild (LFW) (Huang, n.d.) dataset, which contains over 13,000 face images from the internet, to test our face recognition model. Over 5,700 different individuals are featured, with the dataset emphasizing people from various backgrounds, professions, and ethnicities. The dataset includes images with different expressions and lighting, named after the people in them. Given the focus of this project on face recognition, we also implemented a Python script to capture images of our own faces, allowing us to include our database with these additional samples.

## 2.2 Network Architecture

A review of different Face Recognition (FR) methods was conducted in the paper from (Guo & Zhang, 2019). The study concluded that Convolutional Neural Networks (CNNs), particularly supervised Single-CNNs, were the dominant approach. However, it also identified several areas for improvement, including:

- Learning more discriminative deep features
- Fusing different types of face features
- Utilizing efficient metric learning algorithms
- Designing powerful loss functions
- Adopting proper activation functions

While the paper mentioned Autoencoders, which are commonly used in Heterogeneous Face Recognition (HFR) and leverage unsupervised learning, they were deemed unsuitable for this supervised learning project.

After a careful observation in a variety of models we choose a Siamese Neural Network (SNN) (Koch, Zemel, & Salakhutdinov, 2015). Unlike traditional CNNs for image categorization, Siamese networks compare the similarity between two inputs. This makes them well-suited for FR tasks where the goal is to determine if two images depict the same person. Additionally, SNNs consist of two identical subnetworks that share weights, enabling efficient learning with less training data compared to training separate networks for each face. They also offer flexibility in using different loss functions, such as contrastive or triplet loss (Wong, 2021), allowing for customization based on specific similarity metrics. The chosen network's architecture was derived from the paper by (Koch, Zemel, & Salakhutdinov, 2015). To ensure an optimal comparison of different loss functions, the network structure remained consistent across all loss function variations.

## 2.3 Siamese Neural Network

A Siamese Neural Network (Koch, Zemel, & Salakhutdinov, 2015)(SNN) consists of two identical subnetworks, which are mirror images of each other and share the same parameters and weights. Each subnetwork is a sequence of layers that processes one of the two input vectors. The key characteristic of Siamese networks is that the weights are tied; that is, both subnetworks are constrained to always have identical parameters, ensuring that they process their respective inputs in the same way. During training, a Siamese Network learns to differentiate between pairs of inputs that are similar or dissimilar (Figure 2). It is often trained using a contrastive loss or triplet loss function, which encourages the network to output small distances for similar pairs and large distances for dissimilar pairs.

Each input is fed into its respective subnetwork (Figure 1). These inputs are typically pairs of images, texts, or any data that can be compared. As the inputs pass through the subnetworks, features are extracted by the layers. Given the shared weights, identical features will be extracted from identical parts of the inputs. After processing the inputs, the network calculates a distance or similarity metric between the feature vectors extracted from each input. This metric quantifies how similar or different the inputs are to each other. Common metrics include Euclidean distance, Manhattan distance, or cosine similarity. The final output is typically a single scalar value representing the similarity between the two inputs, although it can also be a vector of features that represent the relationship between the inputs.

Siamese Neural Networks are powerful tools for comparison tasks, offering a way to learn similarity in a deep learning context. Their ability to learn complex patterns and similarities between inputs makes them highly effective for a wide range of applications beyond simple image comparisons.

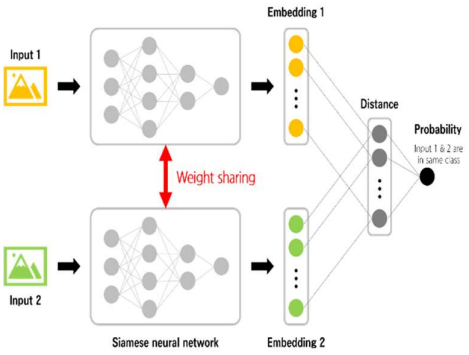


*Figure 1: Concept of an SNN (Yang, 2020)*

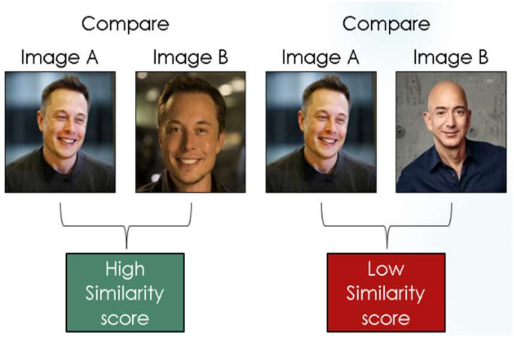


*Figure 2: SNN similar or dissimilar concept (Strahinja, 2021).*

## 2.4 Loss Functions

For our study, we adopted the Siamese Neural Network (SNN) architecture outlined in the paper by (Koch, Zemel, & Salakhutdinov, 2015) incorporating the Binary Cross Entropy Loss (BCE-Loss) as our primary loss function. Additionally, we implemented Contrastive Loss and Triplet Loss, drawing inspiration from the methodologies presented by (Wong, 2021). These choices were made to leverage the strengths of the specified loss functions in enhancing the model's learning capability and performance in our experiments.

### 2.4.1 Binary Cross Entropy Loss

Binary Cross-Entropy (BCE) loss (Godoy, 2018), also known as log loss, is a fundamental concept in machine learning, particularly for tasks involving binary classification. It serves as a performance metric to evaluate how well a classification model distinguishes between two classes. Imagine a model that predicts the probability of an event happening, where 0 represents it not happening and 1 represents it happening. BCE loss measures the discrepancy between these predicted probabilities and the actual labels (0 or 1) in your data.

The formula for BCE loss is:

$$\boldsymbol{L(y,\hat{y}) = -\frac{1}{N}\sum_{k=1}^{N}[y_k \log(\hat{y_k}) + (1-y_k)\log(1-\hat{y_k})]} \quad \textbf{(1)}$$

Where:

- $\boldsymbol{N}$ is the number of observations,
- $\boldsymbol{y_k}$ is the actual label of observation k (0 or 1),
- $\hat{y}_k$ is the predicted probability of observation k
- $\boldsymbol{log}$ is the natural logarithm.

### 2.4.2 Contrastive Loss

Contrastive Loss (CL) (Wong, 2021)is a distance-based loss function used in learning algorithms to distinguish between similar and dissimilar pairs of examples, commonly applied in Siamese networks, one-shot learning, and face verification. Its objective is to pull similar pairs closer and push dissimilar pairs farther apart in the embedding space, based on a defined margin.

The formula for contrastive Loss can be represented as

$$\boldsymbol{L(Y, D) = (1 - Y) * \frac{1}{2} * (D)^2 + (Y) * \frac{1}{2} * \max(0, m - D)^2} \quad \textbf{(2)}$$

Where:

- $\boldsymbol{Y}$ is a binary label indicating if the pair is similar (0) or dissimilar (1),
- $\boldsymbol{D}$ is the distance between the pair of examples in the embedding space, often calculated using Euclidean distance,
- $\boldsymbol{m}$ is the margin, a hyperparameter that defines how far apart the dissimilar pairs should be pushed.

In this formula, the first term penalizes close distances ($\boldsymbol{D}$) for dissimilar pairs (when $Y = 1$), and the second term penalizes distances smaller than $\boldsymbol{m}$ for similar pairs (when $Y = 0$). The margin $\boldsymbol{m}$ acts as a buffer: for dissimilar pairs, if the distance $\boldsymbol{D}$ is greater than $\boldsymbol{m}$, the loss is zero, indicating that the pair is already sufficiently apart. The contrastive loss effectively encourages the model to learn to distinguish between pairs of inputs by adjusting their distances in the learned space.

### 2.4.3 Triplet Loss

Triplet Loss (TL) (Wong, 2021) is a loss function designed to learn distinctions between similar and dissimilar data points in an embedding space. It operates on three data points: an anchor, a positive example (same class as the anchor), and a negative example (different class). The goal is to make the anchor closer to the positive example than to the negative example by a margin.

The loss formula is:

$$\boldsymbol{L(A,P,N) = maxd(A,P) - d(A,N) + \alpha, 0} \quad \textbf{(3)}$$

Where $\boldsymbol{d(A,P)}$ and $\boldsymbol{d(A,N)}$ are distances between the anchor-positive and anchor-negative pairs, respectively, and $\boldsymbol{\alpha}$ is a predefined margin. Triplet loss encourages the model to place data points of the same class closer together and data points of different classes further apart in the embedding space, enhancing the model's ability to differentiate between classes.

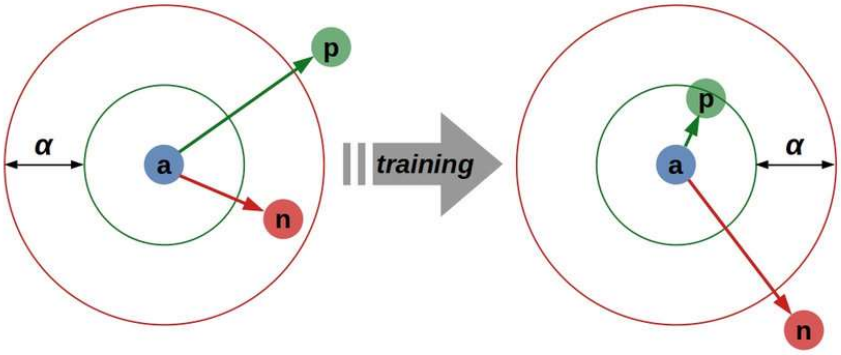


*Figure 3: Visual representation of training a Model with triplet loss. (Matties, n.d.)*

### 2.4.4 Hyperparameters

To achieve optimal comparison across different loss functions, we standardized the hyperparameters setting of the SNN from the paper by (Koch, Zemel, & Salakhutdinov, 2015). This involved setting the number of epochs to **5** and employing the Adam optimizer with a learning rate of $\boldsymbol{1e^{-4}}$. The primary variations in hyperparameters were studied in the Contrastive Loss and Triplet Loss functions, specifically in the margin hyperparameter $\boldsymbol{\alpha}$**.** We experimented with $\boldsymbol{\alpha}$ values ranging from **0.1 to 1**, identified as the optimal margin range for these loss functions, to fine-tune the model's performance.

## 2.5 Activation functions

The ReLU (Rectified Linear Unit) (Gehad, n.d.) activation function outputs the input directly if it is positive, otherwise, it outputs zero. It's widely used for its simplicity and efficiency in training deep neural networks. Mathematically, it is expressed as:

$$f(x) = \max(0, x) \tag{4}$$

The SeLU (Scaled Exponential Linear Unit) (Gehad, n.d.) function, on the other hand, scales and normalizes the outputs for each neuron's activation, aiming to maintain a zero mean and unit variance across layers during training. This property helps in controlling the vanishing and exploding gradients problem, making networks self-normalizing. Mathematically, it is expressed as:

$$\begin{aligned} f(x) &= \lambda x && if\ x > 0 \\ f(x) &= \lambda\alpha(e^x - 1) && if\ x \leq 0 \end{aligned} \tag{5}$$

Where $\lambda$ **(6)** and $\alpha$ **(7)**are the following approximate values:

$$\lambda \approx 1.0507009873554804934193349852946 \tag{6}$$

$$\alpha \approx 1.6732632423543772848170429916717 \tag{7}$$

The SeLU activation function can outperform ReLU by offering more precise performance, primarily due to its self-normalizing property. This unique feature enables SeLU to regulate the mean and variance of the activations across the layers of a deep neural network effectively. This advantage becomes even more significant in tasks were maintaining a consistent level of normalization and preventing dead neurons (neurons that consistently output zero) are vital for optimal performance. When using SELU, there might be less need for batch normalization or other normalization techniques, as the activation function inherently keeps the activations at a controlled scale, simplifying the network architecture and potentially enhancing training efficiency (Gehad, n.d.).

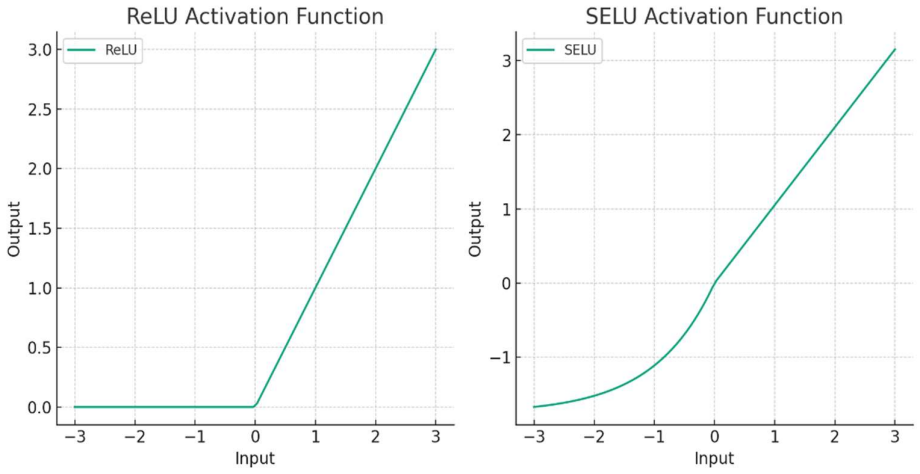


*Figure 4: ReLU and SELU activation functions.*

# 3 Implementation

## 3.1 Dataset handling

As previously mentioned, our dataset consists of the LFW-Dataset (Huang, n.d.) collection and our own captured facial images. Given that SNN (Koch, Zemel, & Salakhutdinov, 2015) require a specific approach to dataset organization namely, segregating the data into negative, positive, and anchor images. We use LFW to subset as our negative dataset. Meanwhile, our own facial images, after undergoing various data augmentation techniques, are divided into positive and anchor sets. This step-in data preparation is crucial, enabling our SNN to effectively differentiate between similar (positive) and dissimilar (negative) images in relation to our anchor images. Such organization ensures that the network can learn the facial similarities and dissimilarities, a key aspect of its training process.

Our project also adapts our dataset to suit three distinct loss functions, each requiring a uniquely structured dataset for optimal performance. The "Vanilla" dataset pairs images for BCE-Loss, using specific paths to combine similar and dissimilar pairs, resizing, and converting them to tensors (Koch, Zemel, & Salakhutdinov, 2015). The "CL" dataset, for Contrastive Loss (Wong, 2021), randomly pairs images from an Image Folder, maintaining class-based pairing balance. Lastly, the "TL" dataset, for Triplet Loss (Wong, 2021), generates triplets from designated positive, anchor, and negative images, ensuring dataset consistency and balance, crucial for learning discriminative features. This methodology ensures an evenly distributed number of samples across training, validation, and test sets for each loss function approach.

## 3.2 Data augmentation

The augmentation strategy remains consistent across all models, involving several steps to preprocess images: converting RGB images to grayscale, resizing to 105x105 pixels, applying random horizontal flips, adjusting saturation randomly, and finally converting the images into tensors (Koch, Zemel, & Salakhutdinov, 2015). This comprehensive augmentation process enhances model robustness by introducing variability and improving generalization.

## 3.3 Network implementation.

The SNN architecture employed mirrors that described in the study by (Koch, Zemel, & Salakhutdinov, 2015). This architecture consists of four convolutional layers, each followed by an activation function, either ReLU or SeLU (Gehad, n.d.), depending on the model variant being analyzed. In addition, the structure includes three max-pooling layers, each with a 2x2 kernel size, a stride of 2, and no padding, to reduce the spatial dimensions of the feature maps progressively. The output of the final convolutional layer is flattened, transforming the feature maps into a one-dimensional vector. The similarity between the compared images is then computed in a subsequent layer, which consists of a linear layer followed by a sigmoid function. This configuration effectively assesses the degree of similarity between input pairs, enabling precise comparison within the SNN framework (Figure 5).

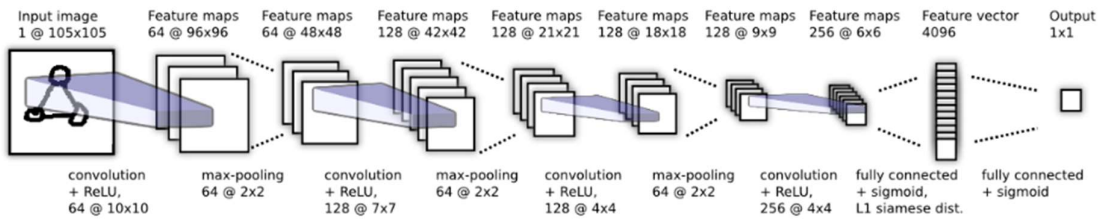


*Figure 5: SNN Framework (Koch, Zemel, & Salakhutdinov, 2015)*

## 3.4 Training Models

### 3.4.1 SNN with BCE-Loss

In this model, we utilize Binary Cross-Entropy (BCE) (Godoy, 2018) Loss, which notably lacks hyperparameters within the loss function itself. Consequently, modifications to the model's performance can only be achieved through alterations in other areas. A significant change implemented was the substitution of the ReLU activation function with SeLU (Gehad, n.d.). This adjustment was made to explore the impact of different activation functions on the models.

### 3.4.2 SNN with CL and TL

These models include a consistent hyperparameter, the margin $(\boldsymbol{\alpha})$**,** across its configurations. To optimize performance, we experimented with different $\boldsymbol{\alpha}$ values, ranging from **0.1 to 1**. In conjunction with this, we also explored the effects of replacing the ReLU activation function with SeLU, to assess the combined impact of these adjustments on the model's behaviour. This exploration led to the generation of 20 unique configurations per model. Given our approach, we have a total of 42 (including the two models of BCE-Loss) distinct models for evaluation, enabling us to test our hypotheses and address our research questions.

## 3.5 Evaluating the Models

In the Section “Material and Methods”, we explored the initial two research questions, focusing specifically on the subsection titled "Network Architecture". Moving forward, the following section”Experimental Results” of the report will concentrate on the remaining three research questions. It's important to note that addressing these questions requires a clear definition of our classification problem, which involves recognizing an input image (anchor image) based on its similarity or dissimilarity. As previously mentioned, we employ the Labelled Faces in the Wild (LFW) dataset (Huang, n.d.), comprising 13,000 images of 5,700 individuals, to serve as the negative image set. In contrast, our own captured facial images will act as the positive set, enabling the identification of our faces in future applications. However, given that each model's loss function operates within a distinct range of values, our analysis will primarily concentrate on comparing the accuracy of each model. This approach is chosen because accuracy provides a normalized metric, allowing for a comparison across the different models.

In our SNN (Koch, Zemel, & Salakhutdinov, 2015) implementation, accuracy is calculated to evaluate the model's performance in distinguishing between similar and dissimilar image pairs. This process involves comparing the model's predictions against the actual labels for each pair of images across the dataset. The model predicts the similarity between pairs of images, outputting a score between 0 and 1 for each pair, indicating the likelihood of the images being similar. A threshold of 0.6 is applied to these prediction scores to classify the image pairs as similar (label 1) if the score is above 0.6, or dissimilar (label 0) if the score is 0.6 or below. For both training and validation phases, the model counts the number of correct predictions (where the predicted label matches the actual label) and divides this by the total number of image pairs to calculate the accuracy for each batch. The accuracies for all batches are averaged to determine the overall accuracy for the training and validation datasets in each epoch. These accuracies are tracked and recorded over the course of the training to monitor the model's performance. Through this method, we effectively measure and analyze the model's ability to accurately classify image pairs as similar or dissimilar, providing a clear indicator of its performance.

# 4 Experimental Results

## 4.1 Which loss functions optimize the performance of SNN?

Upon reviewing the results presented in (Figure 6), a notable similarity is observed between the loss functions of the models.

### 4.1.1 BCE-Loss with ReLU

Focusing on the first set of plots, which would be the SNN with BCE-Loss (Godoy, 2018) approach, we notice that the validation loss closely tracks the training loss, this indicates a robust generalization and an absence of overfitting. Moreover, both the training and validation accuracies for the BCE-Loss rapidly ascend before reaching a plateau, with the validation accuracy marginally trailing the training accuracy. Yet, it remains impressively high, signalling a model that fits well.

### 4.1.2 Contrastive Loss with ReLU

Shifting our attention to the second row of plots, which represent the SNN (Koch, Zemel, & Salakhutdinov, 2015) with Contrastive Loss (Wong, 2021), we observe an increase of the validation loss after the initial decrease, suggesting a case of overfitting. On the other hand, the training accuracy maintains a high level of stability and height, while the validation accuracy, after an initial increase, shows a slight decrease, again indicating overfitting as the model does not generalize well beyond the training data.

### 4.1.3 Triplet Loss with ReLU

The final row offers insights into the SNN (Koch, Zemel, & Salakhutdinov, 2015) with Triplet Loss (Wong, 2021). Like the first model, the training and validation loss both decrease sharply before stabilizing themselves. However, the validation loss consistently exceeds the training loss, hinting at mild overfitting, though less pronounced than that of the Contrastive Loss model. Training accuracy is very high, nearly reaching 1 (or 100%), while validation accuracy, despite being high, shows more fluctuation and at the last epoch does not reach the same level as the training accuracy, indicating a possible overfitting, but still showing good performance.

### 4.1.4 Conclusion

In summary, the SNN with BCE-Loss appears to offer the most generalizable model, with the Contrastive Loss model being the most prone to overfitting, and the Triplet Loss model showing signs of overfitting, yet retaining a high validation accuracy. These patterns suggest that the hyperparameters of the Network might not have been optimally adjusted during training if they remained unchanged. Additionally, it's worth noting that both the Contrastive Loss and Triplet Loss models begin with a lower initial loss compared to the BCE-Loss model, hinting at a more refined learning process.

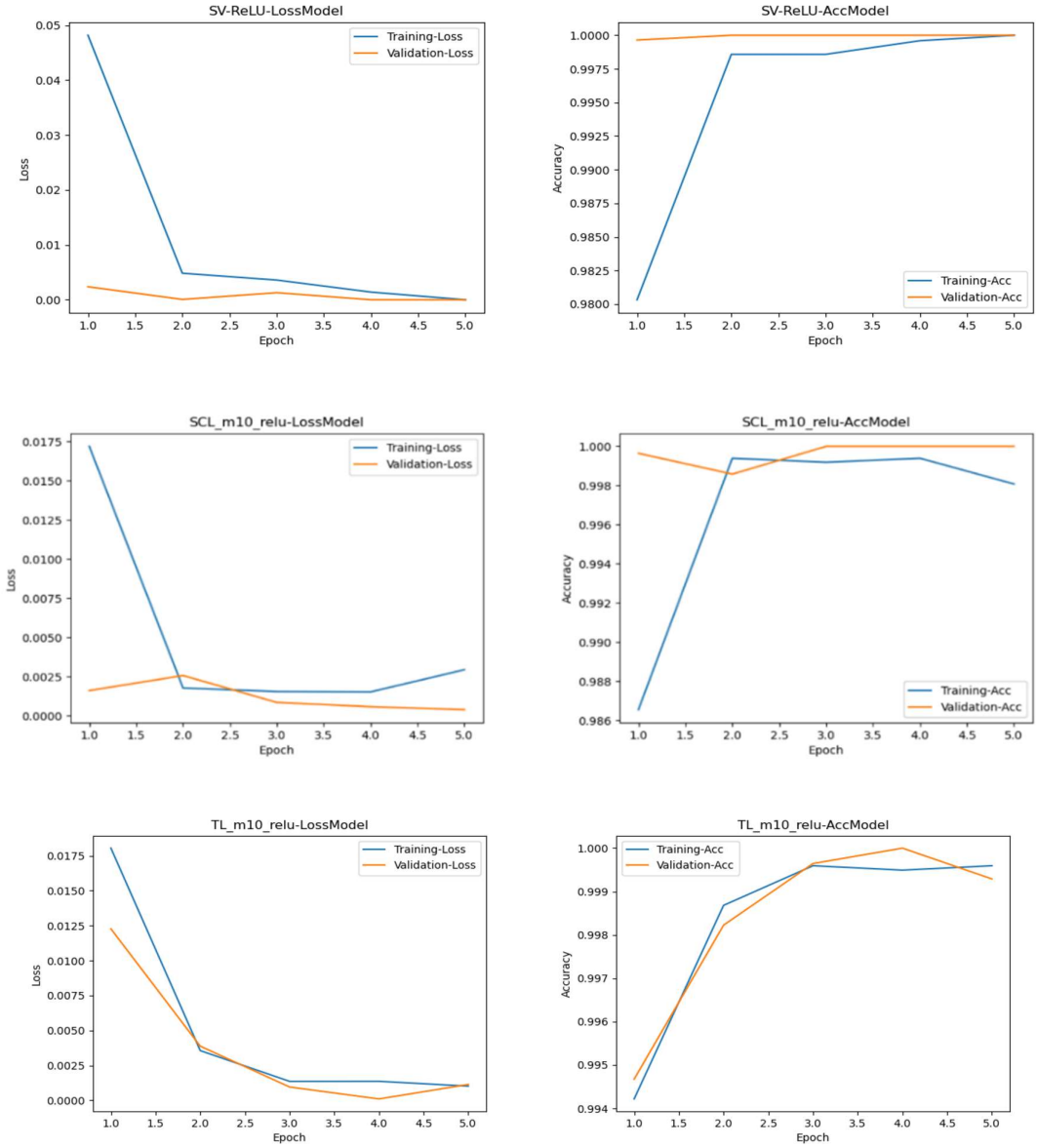


*Figure 6: SNN with BCE-Loss, CL and TL and ReLU*

## 4.2 Which Activation function optimizes the performance of SNN?

### 4.2.1 BCE-Loss with SeLU

Let's start with the SNN (Koch, Zemel, & Salakhutdinov, 2015) using BCE-Loss (Godoy, 2018) (Figure 7), we observe that the training loss decreases sharply before leveling off. The validation loss initially mirrors the training loss yet slightly increases at the end of the training epoch. Training and validation accuracies both increase rapidly and then reach a plateau, with training accuracy surpassing minimal the validation accuracy. The model demonstrates high accuracy levels, with a slight discrepancy between the training and validation results. Notably, the loss begins around 0.030, which is a more favourable starting point than that observed with ReLU. Switching from ReLU to SeLU in the BCE-Loss model does not lead to a dramatic change in performance. However, the SeLU model appears to be more stable, and when compared with the ReLU model (Figure 3), it seems to have mitigated the minor gradient issues that were present mid-training of the ReLU model.

### 4.2.2 Contrastive Loss with SeLU

In the SNN (Koch, Zemel, & Salakhutdinov, 2015) utilizing Contrastive Loss (Wong, 2021) (**Error! Reference source not found.**) with the SeLU activation function, the training loss decreases steadily, not as dramatically as in other models, but it stabilizes in a consistent manner, suggesting reliable model performance. Compared to other models, however, the decrease in loss during training and validation is less pronounced, suggesting that the model may require additional epochs to achieve loss values comparable to those of the BCE-Loss or Triplet Loss models. Additionally, the training and validation accuracies are lower than those observed in the other approaches. Furthermore, a point of concern is that the validation accuracy exhibits a decline at the end of the epoch, which may indicate overfitting. Swapping ReLU with SeLU in the Contrastive Loss model again enhances the model's stability, which makes the model stronger against overfitting. Nevertheless, this configuration seems to need a greater number of epochs to reach a satisfying level of loss and accuracy.

### 4.2.3 Triplet Loss

This method shows the most rapid initial reduction in training loss among all the models evaluated. The SNN (Koch, Zemel, & Salakhutdinov, 2015) employing Triplet Loss (Wong, 2021) (Figure 7) demonstrates promising signs by plateauing after the sharp decrease. However, a slight increase in validation loss after the third epoch hints at the start of overfitting to the training data. While training accuracy approaches near-perfect levels, validation accuracy experiences a slight drop towards the later epochs, further suggesting overfitting might be a concern. When we compare this SeLU-based Triplet Loss model with its ReLU (Figure 6), the SeLU model appears to again offer improved stability and seems to be more robust against overfitting.

### 4.2.4 Conclusion

This section of the project provides valuable insights into behaving of the SeLU over ReLU. SeLU can be considered better than ReLU in certain contexts because it has a self-normalizing property that helps maintain a mean and variance close to 0 and 1, respectively, through the layers of a deep neural network. This promotes a stable and faster convergence during training, what we could clearly saw in each of our different models. Furthermore, it's worth noting that SeLU's impact on model performance is not uniform across all loss functions. Each loss function reacts uniquely to the SeLU activation, highlighting the importance of fine-tuning hyperparameters to optimize the network's behaviour.

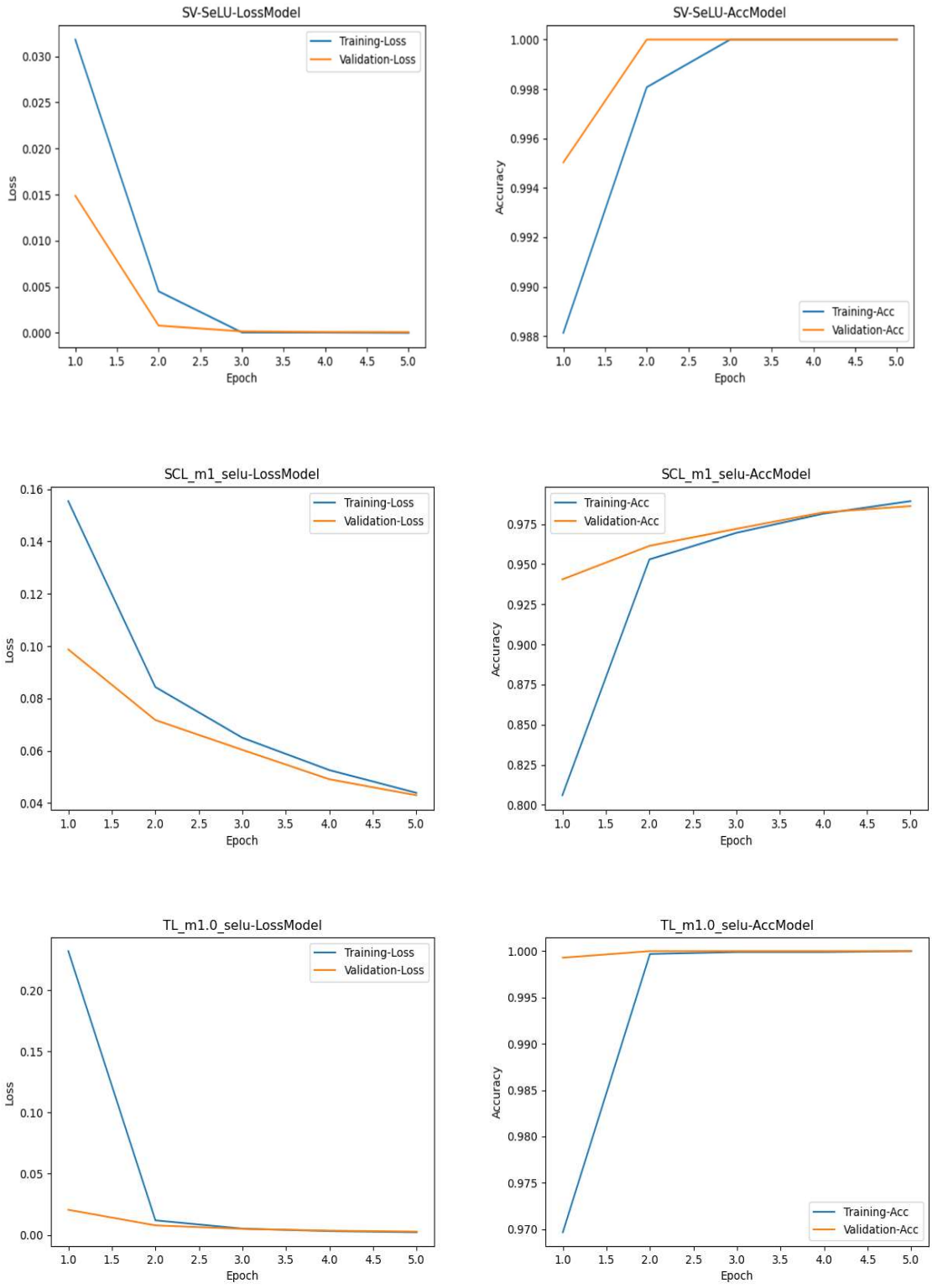


*Figure 7: SNN with BCE, CL and TL and SeLU*

## 4.3 How does hyperparameter tuning impact the SNN's accuracy and efficiency?

### 4.3.1 Contrastive Loss

Upon examining the loss trends (Figure 8), it is evident that increasing the margin hyperparameter value in Contrastive Loss (CL) (Wong, 2021) leads to a sharper initial decrease in loss. This suggests that a larger margin can contribute to a steeper gradient descent at the onset of training. Additionally, when comparing activation functions, models using the SELU activation function exhibit an earlier and sharper decline in initial loss with a smaller margin compared to those using ReLU. This could indicate that SELU is more sensitive to margin adjustments in the early training stages. Despite this, towards the latter stages of training, it is observed that models with SELU activation end up with higher loss values compared to their ReLU counterparts. This might imply that SELU-activated models could benefit from either an increased number of epochs or a higher learning rate to match the performance of ReLU models. Nonetheless, across all configurations is the plateauing of the loss as epochs advance, hinting at a potentially suboptimal learning rate that is too low to further learn in loss towards the end of the training process.

Turning our attention to the accuracy plots (Figure 9), both SELU and ReLU (Gehad, n.d.) models appear to struggle with a lower margin value. The divergence in performance becomes more visible at a margin threshold of 0.5, with ReLU models outperforming SELU in terms of accuracy. Interestingly, as the margin value surpasses 0.7, SELU models demonstrate a higher initial accuracy, especially in training results. This could be indicative of SELU's ability to capitalize on higher margin values more effectively during the initial training phase.

To enhance the performance of these models, several strategies could be considered. Firstly, for SELU models that plateau early with higher loss values, increasing the learning rate or the number of training epochs may help in achieving further loss reduction. Secondly, since both activation functions show difficulties with lower margin values, it would be beneficial to experiment with varying margin values to find an optimal point that balances the initial learning boost and the ultimate accuracy. Lastly, given that all models exhibit plateauing behaviour, a dynamic learning rate schedule could be introduced to maintain the momentum of loss reduction throughout the training duration.

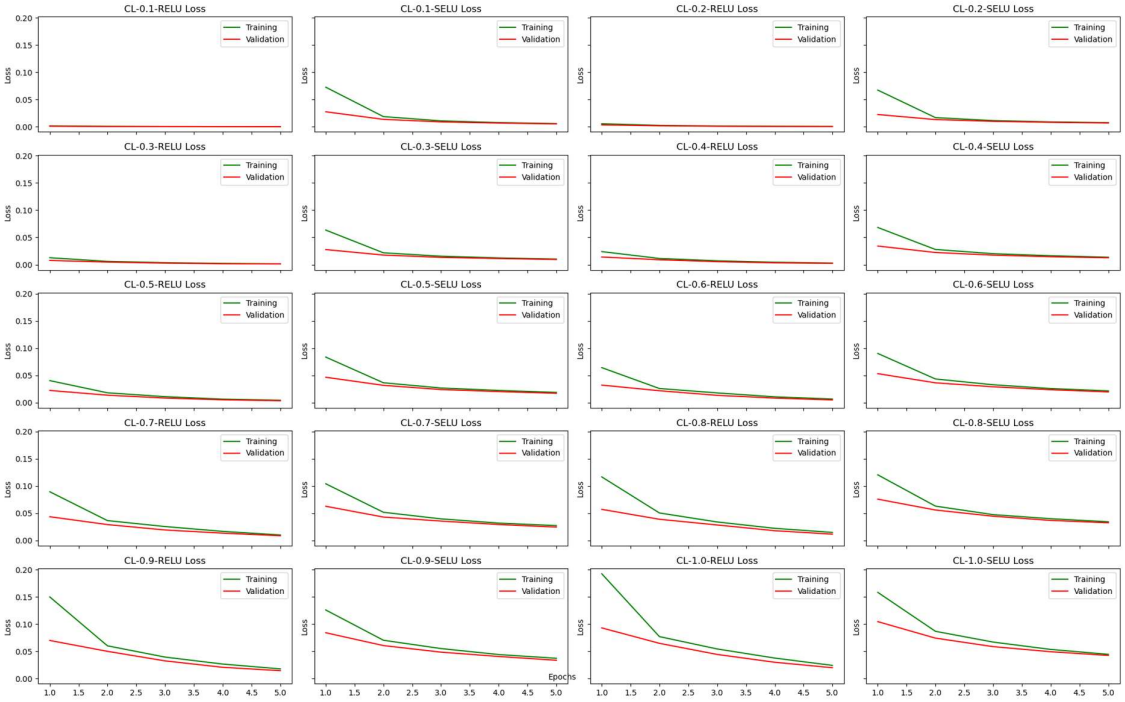


*Figure 8: CL-Loss with different margins.*

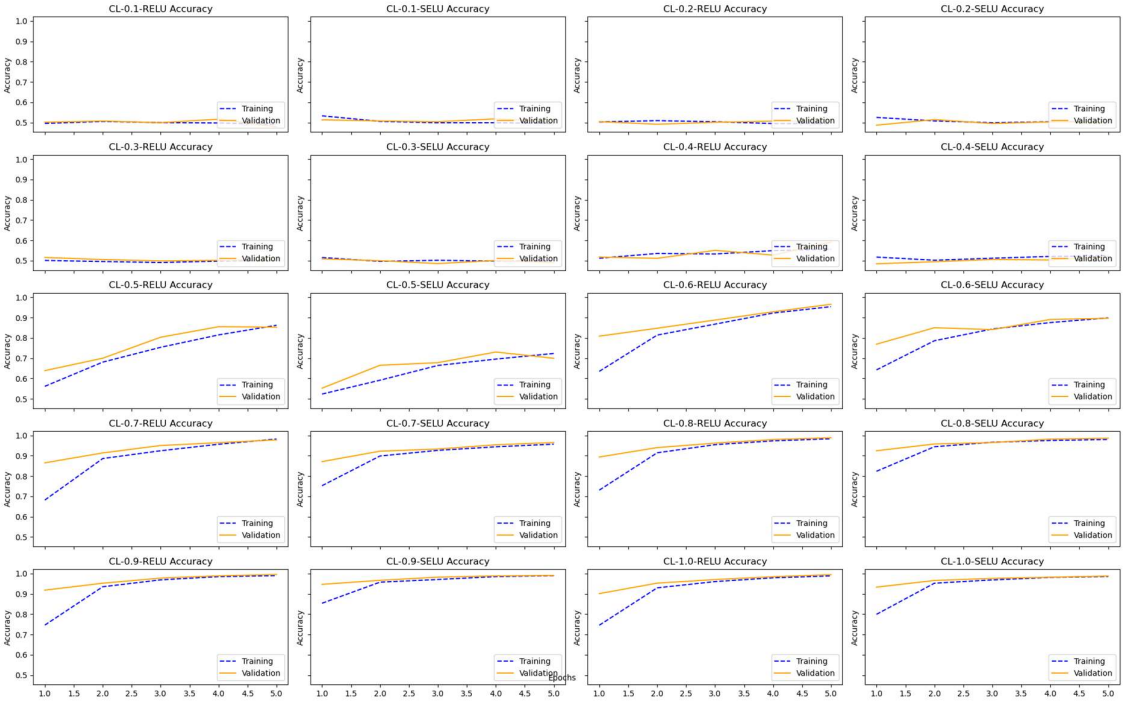


*Figure 9: CL-Acc with different margins.*

### 4.3.2 Triplet Loss

The examination of the results for Loss (Figure 10), obtained with Triplet Loss (TL) (Wong, 2021) proofs our previous findings from earlier analyses. Notably, an increase in the margin of the loss function leads to a sharper initial decrease in loss across all model configurations. This observation highlights the significance of the margin as a critical hyperparameter in determining the loss function's impact. It is observed that models using the ReLU activation function begin training with a higher initial loss compared to those using SELU. Nonetheless, a plateau in both training and validation losses becomes evident after the second epoch across all models. This plateau is a strong indicator of a potentially suboptimal learning rate. Without adjustments, the models struggle to learn effectively beyond the initial phases. To improve the learning process, a more precise adjustment of the learning rate is necessary. Such adjustments should aim to ensure a consistent reduction in loss after the initial epochs.

Turning our attention to the accuracy plots (Figure 11), a similar trend is observed as with the loss plots. There is a marked initial increase in training accuracy for each model. The accuracy gap between models using SELU and ReLU (Gehad, n.d.) activation functions is more pronounced and remains consistent despite changes in hyperparameters, indicating a significantly greater impact than observed in the loss plots. However, both training and validation accuracies begin to plateau early, suggesting not only a poor learning rate but also the potential onset of overfitting.

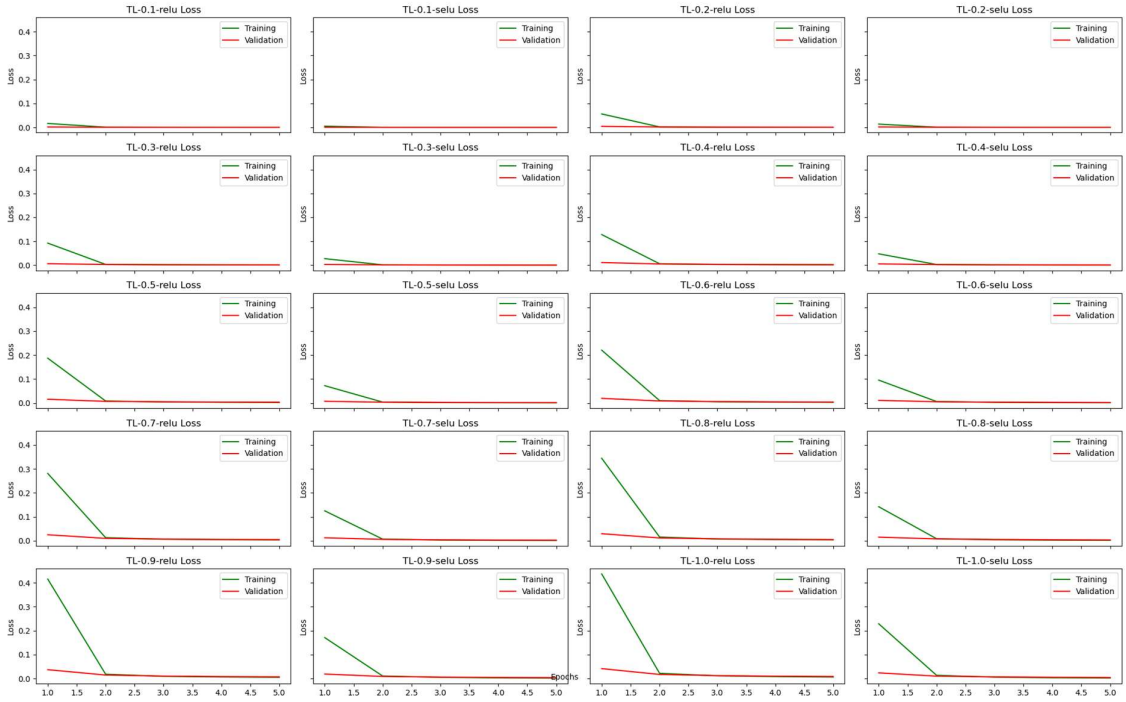


*Figure 10: TL-Loss with different margins.*

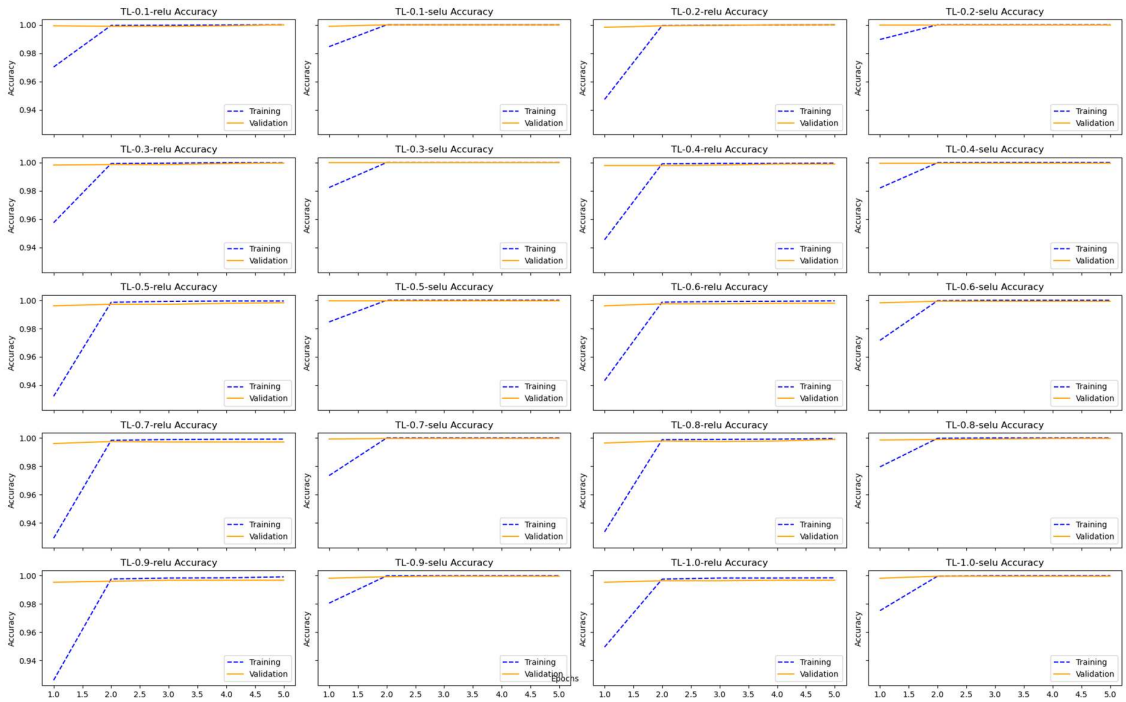


*Figure 11: TL-Acc with different margins.*

# 5 Conclusion and Outlook

To get to the conclusion and outlook for this project on face recognition using neural networks we want to highlight again our research questions and then answer that we found via this project.

**Overview of the research questions:**

1. Which Neural Network architecture demonstrates the most effective performance for FR tasks?
2. What are the unique strengths of SNN?
3. Which loss functions optimize the performance of the chosen network architecture for FR application?
4. Which Activation function optimizes the performance of the chosen network architecture for FR application?
5. How do different hyperparameters settings within the chosen loss function impact the network's accuracy and efficiency?

## 5.1 Question 1

This question has already been addressed in the "Network Architecture" subsection of our report. However, it should be mentioned again that in this project, we explored only a fraction of the various approaches to this classification problem. Despite the limited scope of our investigation, it is noteworthy that the Single-CNN architecture is widely recognized for its applicability in face recognition tasks. Nevertheless, the Siamese Neural Network (Koch, Zemel, & Salakhutdinov, 2015), representing a multi-CNN approach, demonstrates immense potential for such applications.

## 5.2 Question 2

This question is comprehensively addressed in the "Siamese Neural Network" subsection of our report. For a detailed exploration of this topic, we kindly invite the reader to refer to the named section.

## 5.3 Question 3

By completing this project and analyzing various loss functions, we have found that Triplet Loss stands out as a highly effective choice for face recognition (FR) problems. This loss function delivered impressive performance with minimal hyperparameter tuning, affirming its suitability for FR challenges. However, it's worth noting that despite underperforming compared to Binary Cross-Entropy (BCE) Loss and Triplet Loss (TL), Contrastive Loss should not be overlooked. Its robustness against overfitting marks it as a potentially valuable option in scenarios where generalization is critical.

## 5.4 Question 4

Our experiment of activation functions has shown that the Scaled Exponential Linear Unit (SeLU) may offer a more stable and robust performance compared to the Rectified Linear Unit (ReLU). Its benefits are especially notable in managing gradient-related challenges and promoting quicker convergence. Based on these insights, using SeLU into neural network architectures appears to be a beneficial strategy for enhancing model performance. By doing so, it unlocks the models' latent capabilities, potentially leading to higher accuracies and reduced loss values.

## 5.5 Question 5

Hyperparameter tuning, particularly the adjustment of margin values in loss functions, was identified as crucial for improving the SNN (Koch, Zemel, & Salakhutdinov, 2015) accuracy and efficiency. A more dynamic approach to learning rate adjustment and an increase in training epochs or learning rate for models using SeLU might address some observed plateauing in loss reduction.

## 5.6 Outlook

The outlook for future research suggests exploration of hyperparameter optimization, especially learning rates, optimizer, and margin values, to enhance model performance. Additionally, investigating other architectures or advanced techniques might provide new insights into improving face recognition technologies.

# 6 References

Gehad, A. S. (n.d.). *opengenus.org*. From https://iq.opengenus.org/scaled-exponential-linear-unit/

Godoy, D. (2018, November 21). *https://medium.com/*. From https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a

Guo, G., & Zhang, N. (2019). *Computer Vision and Image Understanding.* Elsevier.

Huang, G. (n.d.). *Labeled Faces in the Wild*. From Labeled Faces in the Wild: http://vis-www.cs.umass.edu/lfw/

Koch, G., Zemel, R., & Salakhutdinov, R. (2015). *Siamese Neural Network for One-shot Images Recognition.*

Matties, M. (n.d.). *www.researchgate.net*. From www.researchgate.net: https://www.researchgate.net/profile/Mark-Matties/publication/346030429/figure/fig1/AS:959937517740035@1605878347478/Illustration-of-the-triplet-loss-constraint-showing-three-embedding-samples-anchor-blue.jpg

Strahinja, S. (2021, 11 07). *Datahacker.com*. From https://datahacker.rs/019-siamese-network-in-pytorch-with-application-to-face-similarity/

Wong, L. (2021, March 31). *Lil'Log*. From Lil'Log: https://lilianweng.github.io/posts/2021-05-31-contrastive/

Yang, T. (2020, October 20). *tyami.github.io*. From https://tyami.github.io/deep%20learning/Siamese-neural-networks/

# 7 List of Figures