TECHNION
Israel Institute of Technology

The Andrew and Erna Viterbi Faculty of
ELECTRICAL & COMPUTER ENGINEERING

Laboratory of ComputerGraphics & Multimedia
Department of Electrical Engineering Technion
all that meets the eye...

044167 Project A – Project Portfolio

Lab Name	Computer Graphics and Multimedia
Project Name	Backward Compatibility in Face Recognition
Project ID	6461
Semester	Winter 2021/2022
Supervisor	Elad Hirsch
Students	Shai Yehezkel 205917883
	Lior Dvir 207334376

Table of Contents

Background

To get familiar with notations and symbols we use in our work, this section is dedicated for going through preliminaries and concepts that will be in our project.

Machine Learning

The study of computer algorithms that can improve automatically through experience and using data[1].

In our scope data will be images of human faces. At some point we will be handling embedding vectors, a type of data which is extracted from images.

Our training is being conducted in a **Supervised Learning** fashion, E.G our input data is always labeled as we know what the desired output is.

Neural Network

One of many ways to solve optimization problems is with the use of parametrized models. A state-of-the-art model is a neural network which is widely used in computer vision and machine learning tasks.

A neural network has an output and possibly multiple outputs.

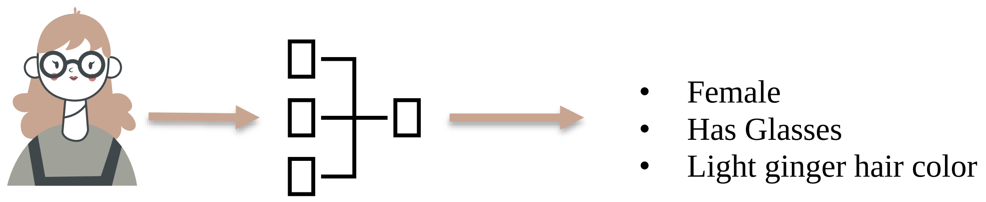

Figure 1 - Neural Network

[1] https://en.wikipedia.org/wiki/Machine_learning

Model Architecture

Neural networks differ in the number of layers, dimensions of each layer, activation functions and loss functions. The set of these define a specific model architecture.

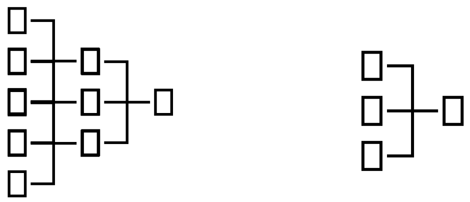

Figure 2 - Model Architectures

Note that when a transformer is addressed in this portfolio, we do not regard to the architecture family of transformers with self-attention mechanisms, but rather a model which is fed an input, and outputs data with manipulations done within.

Embedding Vector, Embedding Space, Manifold

An embedding vector is the output of the last layer of a neural network **before** performing any task (E.G classification). The vector represents deep features and is of can be of any size desired (as a model's parameter).

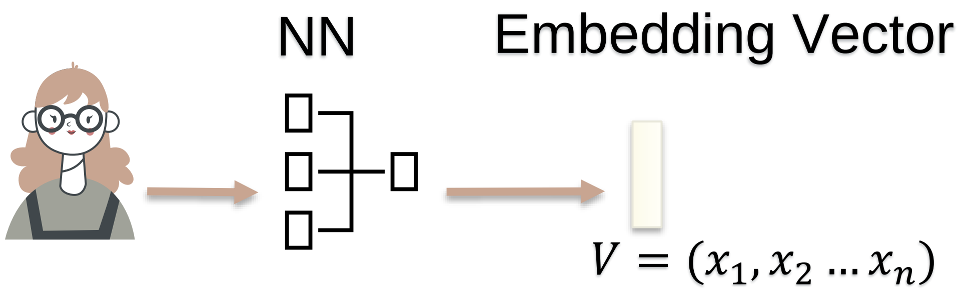

Figure 3- Embedding Vector

In our project we define the embedding vector size to be 512, that is each RGB image will be represented a deep feature vector of size 512.

When looking at multiple images, we can imagine a multidimensional space spanned by the multiple embeddings, each correlate to a single point in that space. In literature it might be called Deep Feature Space.

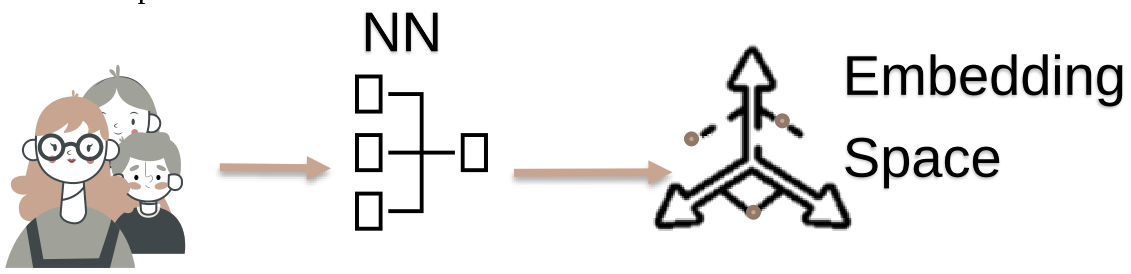

Figure 4 - Embedding Space

Note every network might have a different embedding space.

As this space is not fully sampled, we can imagine a hyper-surface representing out samples in the embedding space called a manifold. Every sample from the dataset has a corresponding representation in the n dimension space sampled in the manifold.

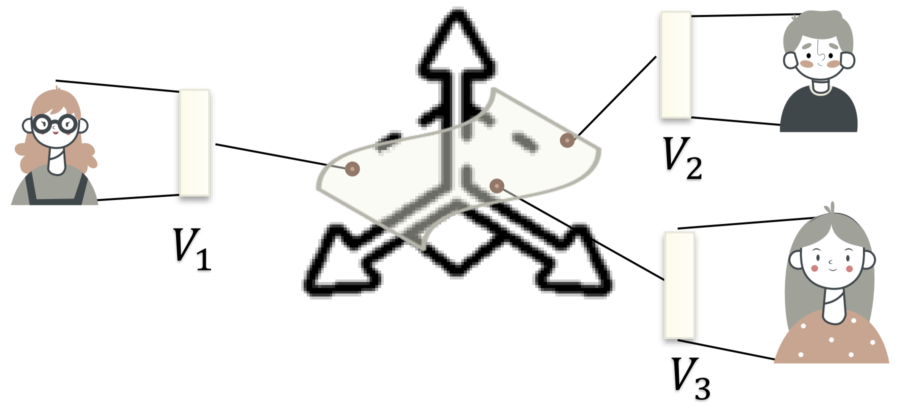

Figure 5 - Manifold

Recognition and Retrieval

As said in previous section, a dataset of images will span an embedding space with specific neural network. The space can be saved into a gallery of known images of people.

As far as privacy issues concern, the gallery consists of embedding vectors, and not of images.

When a specific image is fed-forward through the network, we can define the process of querying the gallery with an embedding vector as retrieval.

A vector is associated with a unique person, and thus a recognition process is achieved by finding the closest match to it in the gallery and returning the label associated.

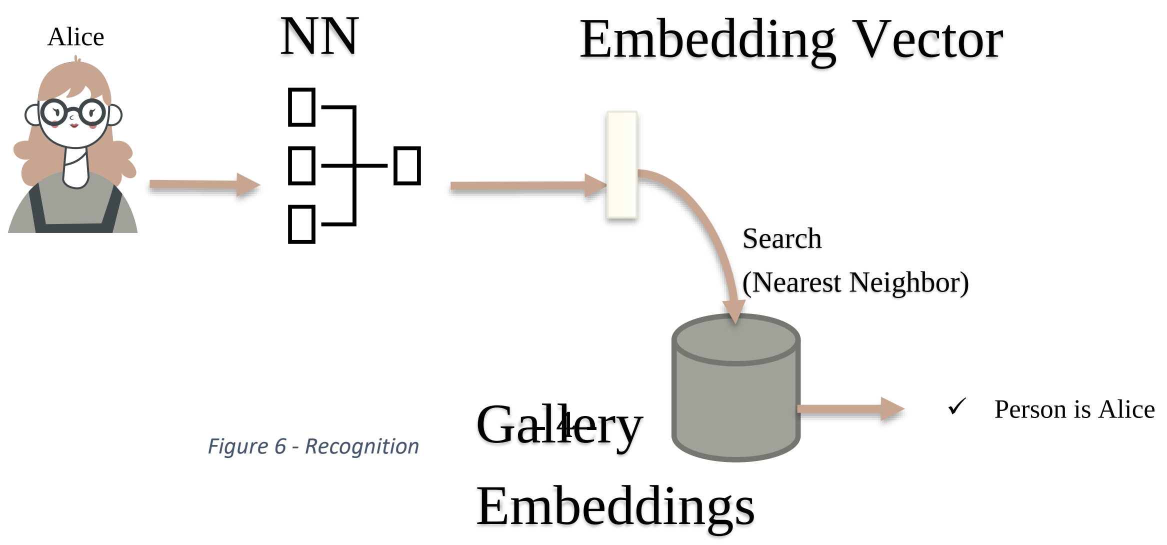

Figure 6 - Recognition

Project Definition

Our work is based on the following paper: Unified Representation Learning for Cross Model Compatibility

This section is dedicated for explaining what the Cross Model Compatibility problem is, what is our project scope, and what we propose for enhancing existing approaches.

Cross Model Compatibility

Visual recognition and retrieval systems are widely used in out lives. Examples for use are frictionless physical access systems such in Apple ID, missing persons search, global place recognition and even homeland security applications.

et us recall the standard process of recognizing a person – that is feeding forward an image through the network, receiving its embedding vector representation, and with that vector query an existing gallery of embeddings associated to a specific network. The label of the closest match the embedding vector will be the person's ID.

Let:

- $NN\#1$ be a neural network which extracts deep features
- A the spanned embedding space by $NN\#1$.

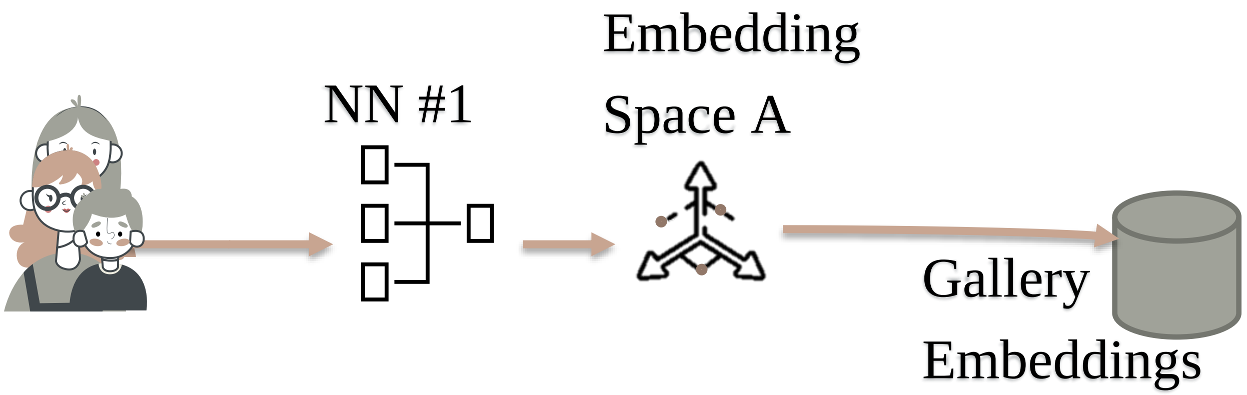

Figure 7 - Standard Retrieval System

As research progresses, more deep network architectures are discovered, ones with potentially a better performance than $NN\#1$. Let $NN\#\ 2$ be a new architecture which improves our current retrieval system, in the sense that the percentage of people that are recognized (or not recognized when they do not exist) is higher.

Recall each network spans a different embedding space which might differ both in dimensions and the manifold of our samples.

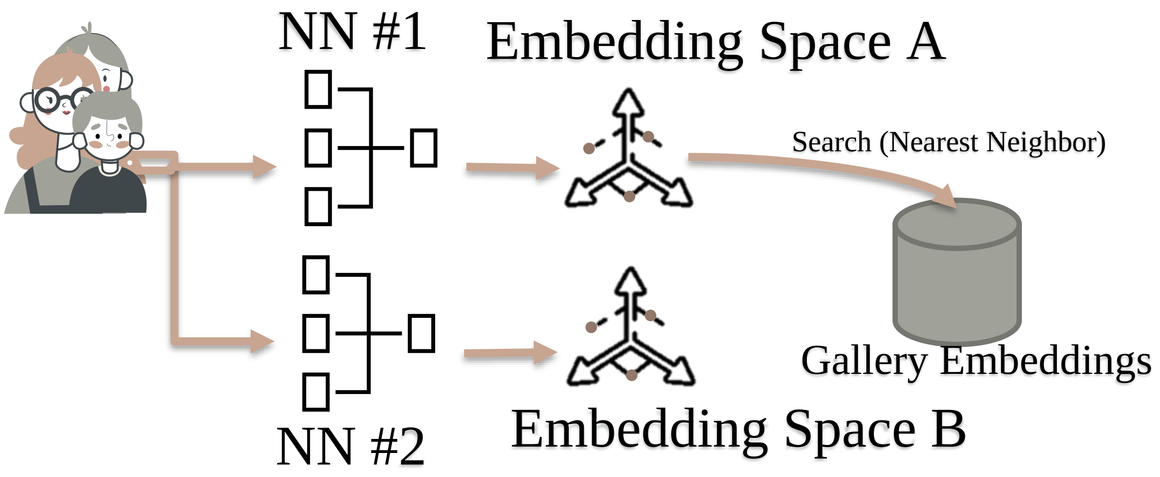

Figure 8 - Second NN

We would like now to transfer our work to the new architecture, so that every image now will be fed to $NN\#2$. However, querying directly from B the old gallery embeddings will probably yield low accuracies as the manifold differs.

Addressing how a new neural network with a new embedding space can retrieve valid information from the old gallery embeddings is what is called the Cross Model Compatibility.

Naïve Approach

A naïve approach to solve this problem will be taking the entire dataset and feeding it through $NN\#2$, and create a new gallery embedding.

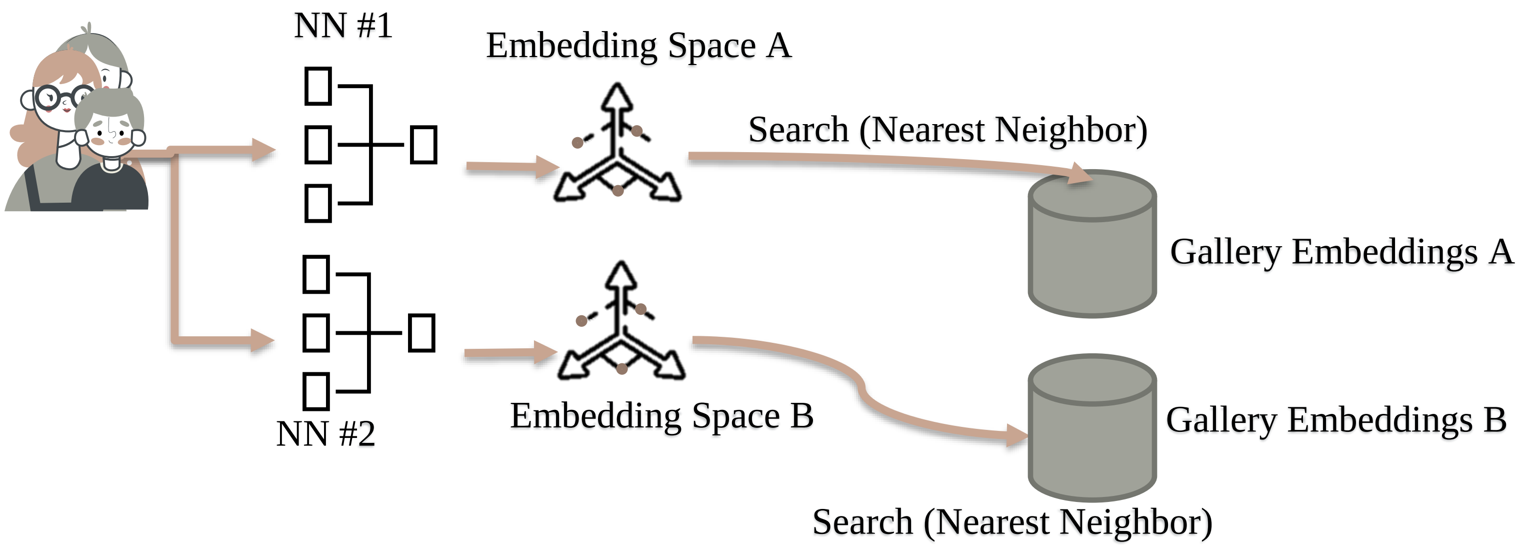

Figure 9 - Naive Approach

However, this approach will require re-embedding all existing images of identified people. This process can take a long time, and moreover – it imposes privacy issues as this will require saving images of people, and not only their embedding vectors.

Unified Transformer

A thorough work proposed by Wang et al. involves creating a unified transformer which is trained together but transforms independently $Gallery\ A$ and Embedding Space B into $Gallery\ B$ and embedding space A'.

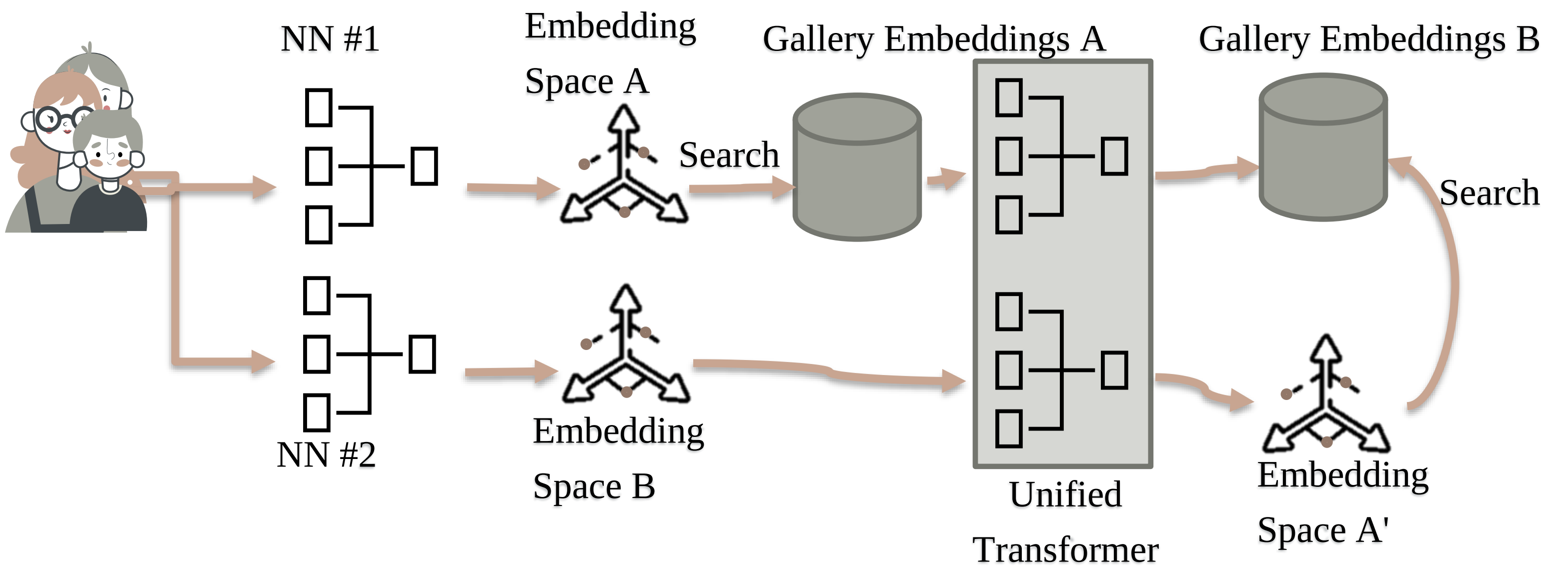

Figure 10 - Unified Transformer

This approach optimizes a new and unified embedding space A' and leverages information both from the original gallery A and new embedding space B. In our work decided to focus on a different approach also reviewed in the mentioned paper.

Embedding Space Transformer

Another approach which might yield satisfactory results would be training a transformer which transforms embedding space B into the gallery embedding space.

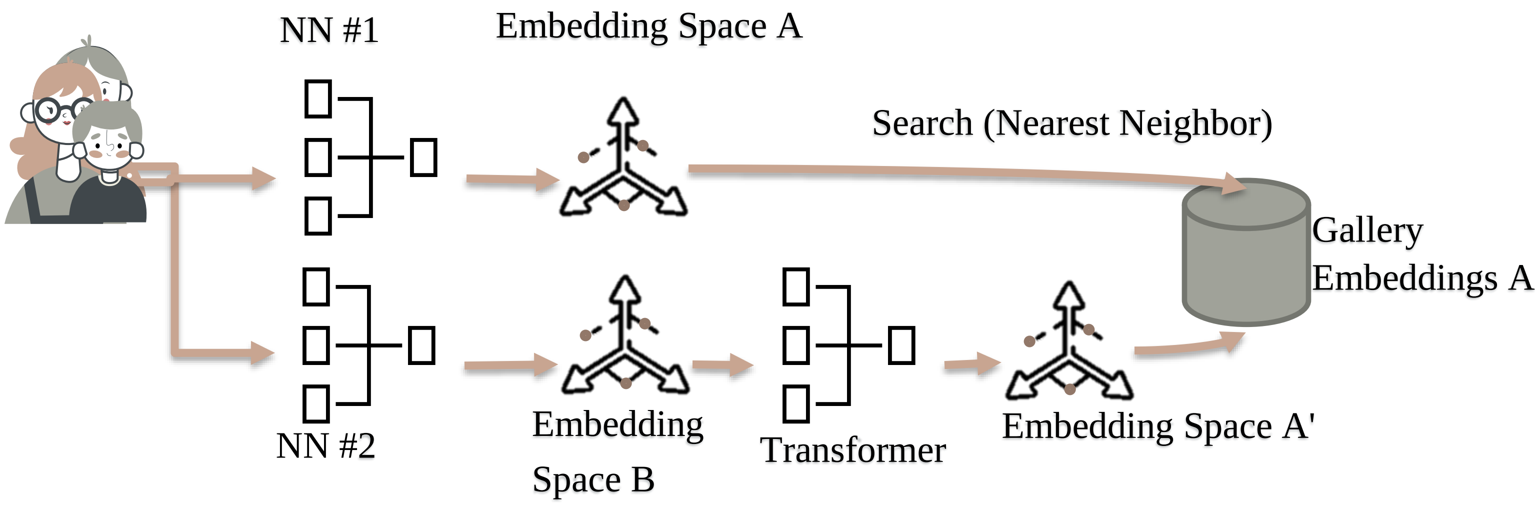

Figure 11 - Space Transformer

In our work we implemented this solution and added enhancements which improved the accuracy of recognition after feeding $NN\#2$.

Proposed Improvements

To the existing transformer we will add new loss functions to the optimization process.

Moreover, we will perform interpolation of embedding vectors resulting in new samples.

A good interpolation is a one which will create new samples which are on the manifold of A', thus introducing new information into the system.

Interpolation methods and loss functions will be discussed in next sections.

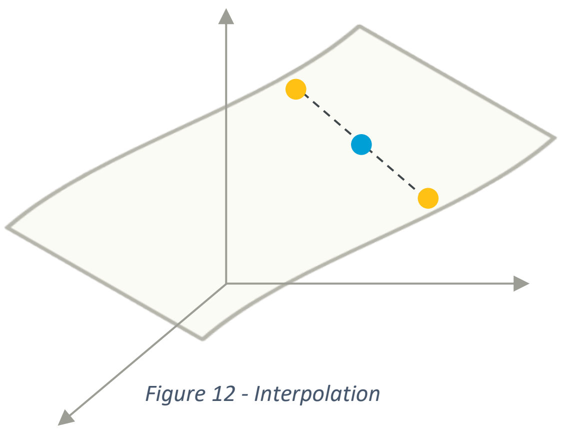

Figure 12 - Interpolation

Assumptions

To simplify our task and to adapt the Cross Model Compatibility problem to the time scope of out project, a few basic assumptions were made and were taken into consideration in our work.

Our work is based on recognition systems in the InsightFace repository.

The dataset we are working with is *MS1M-ArcFace* which holds 5.8 million RGB images of almost 90,000 different people (labels), where In this dataset, all images are visible, centered and aligned.

Moreover, we will assume the networks that will serve as $NN\#1$ and $NN\#2$ will perfectly label all existing images in the dataset, as we do not consider any loss of performance incurred by miss-classifying them. This assumption turns to be quite accurate as we trained two backbones with accuracies of 96%-98% in labeling images.

Lastly, we do not limit the number of parameters used by our transformer as this allows us to explore different architectures and focusing on the improvements we introduced.

Challenges

Before starting our project, we tried identifying some of the possible difficulties we might face. These include:

Adjustments of InsightFace repository – The repository is large and consists of 2M+ lines of code. Finding the correct modules that we will benefit from and changing some of its code in the embedding vector manipulation.

Computation Limitations – As we process a large amount of data, and we are pre-processing many embedding vectors before even training our transformer. Moreover, training network, both InsightFace's networks and our transformer might take a long time.

Limited Timeframe – As this is our project for submission, the extent in which we will be able to improve our transformer is limited.

Work Done

Training ResNet18 and ResNet50

ResNet18 and ResNet50 models were taken from InsightFace repository and act as the old face recognition model and the improved model respectively.
We trained both models with the ArcFace dataset also taken from the same repository with them reaching classification success rate of about 96%.

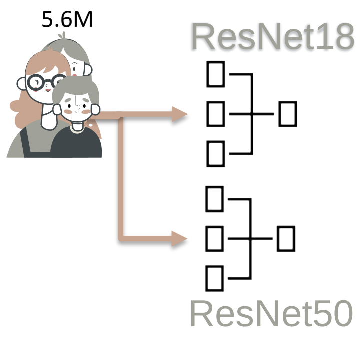

Figure 13 - Training ResNet18 and ResNet50

Creating the New Dataset

ResNet architecture is composed of a series of blocks containing convolutional layers and identity layers (also called residual blocks) followed by Fully Connected (FC) layers and finally a Softmax layer which outputs the classification. We extract the output of the final FC layer (right before the Softmax) and save it as the Deep Features Vector of the image, also called the Embedding Vector. Each Embedding Vector is of 512 dimensions and together all the Embedding Vectors spread the Embedding Space.
We do the extraction for the whole dataset twice, once for ResNet18 and once for ResNet50. The Space spread by ResNet18's Embedding Vectors represents the old Gallery space, while the space spread by ResNet50's Embedding Vectors represents the new Query space.

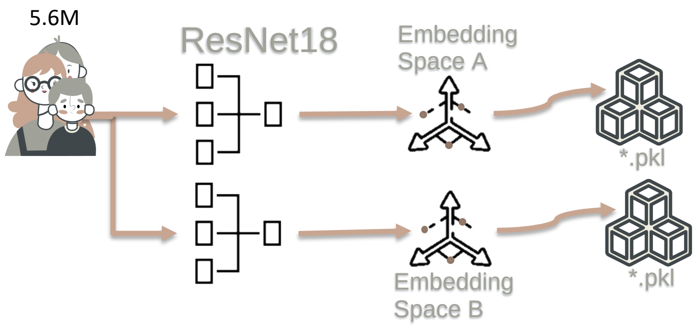

Figure 14 - Extracting Embedding Vectors from RN18 and RN50 and storing them in PICKLE files

We first attempted to save the new embedding data using Pandas and CSV files, however the data proved too large for out limited memory space. We then moved to a compressing approach and chose PICKLE (A compression module for Python). It worked but garnered new challenges; We had to develop a new data-frame especially (Including a loader and a sampler) as well as split the data into multiple files.

Loss Functions

As part of the training process for the Transformer we've decided to use two different loss functions.

The first is Cosine Similarity Loss (CSL) which represents the cosine of the angle between two vectors. It returns a value in the interval [-1,1]. 1 being perfectly parallel and -1 being perfectly anti-parallel. (We shift this number by calculating *1 – CSL* to conform to "smaller loss is better" approach and get 0 for perfect parallel and 2 for perfect anti-parallel)

The second is Mean Square Error (MSE) loss which represents the element-wise squared distance between two vectors.

Cosine Similarity Loss: $$L_1 = \frac{\mathbf{A}\cdot\mathbf{B}}{||\mathbf{A}||\cdot||\mathbf{B}||}$$

MSE Loss: $$L_2 = \left||A - B\right||_2^2$$

Figure 15 - Mathematical formulas for CSL and MSE

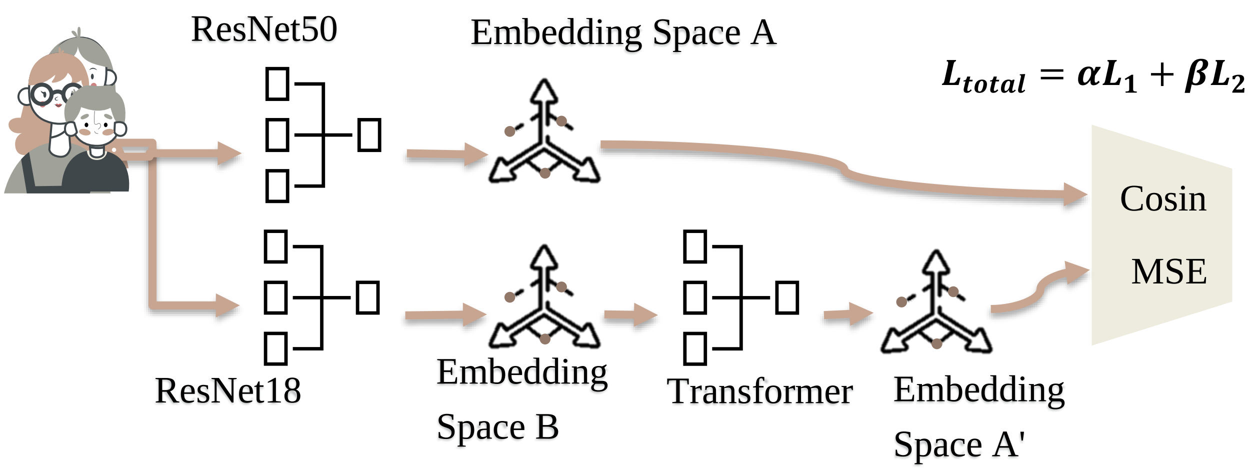

Figure 16 - Using multiple loss functions

Out first approach was to weight both loss functions the same as such $L_{total} = \frac{1}{2}CSL + \frac{1}{2}MSE$
This way both contribute equally to the training process.

However, the actual results corresponded to $L_{total} \approx \frac{1}{2} MSE$, which meant the MSE contribution overwhelmed the CSL contribution. Therefore we weighted the functions accordingly:

$L_{total} = \alpha \cdot CSL + \beta \cdot MSE,\ where\ \alpha \gg \beta$

After tweaking the weights we received better accuracy results than using only one function.

Transformer Architecture

We have experimented with two different architectures for our Transformer.

1) MLP

   Chosen as the baseline architecture, it contains a single Fully-Connected 512 to 512 layer

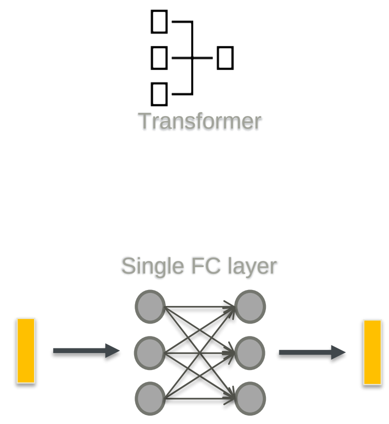

Figure 17 - MLP containing a single FC layer

2) MSResNet

   Multi-Scale ResNet. It contains three different branches of different scaled 1D convolutional layer (3/5/7) which are concatenated into a FC layer. The architecture is designed for Deep Features Vectors. We expected it to outperform the MLP, although we later observed the MLP produce better results which led us to drop the MSResNet model and continue with the MLP only.

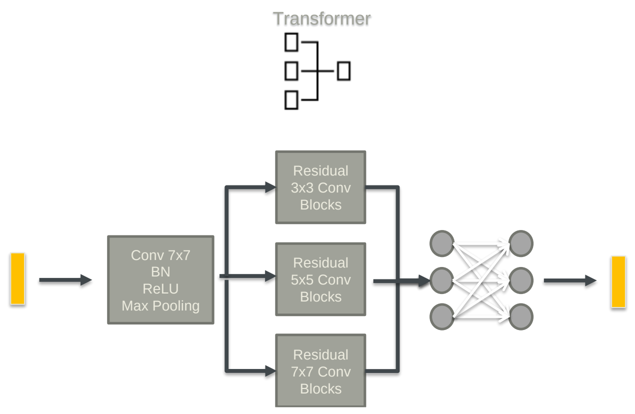

Figure 18 - Multi-Scale ResNet model

Interpolation of Embeddings

An interpolation of two Embedding Vectors is the weighted sum of them. In our case we weighted them equally, so it is the mean of the vectors.
By interpolation multiple embedding samples of existing two labels, we create a new label containing the interpolated embeddings of the two.
The idea behind the interpolation is like so:
Let's assume we have embedding vectors A and B in the Query space and their counterparts A' and B' in the Gallery space. C is the interpolation (and in our case the mean) of A and B. and C' is the interpolation of A' and B'. If we assume a somewhat linear transformation between the Query and the Gallery them, we expect that C should be transformed closely to C'. Using this exact assumption, we aim to improve our accuracy through training on these interpolated embeddings.

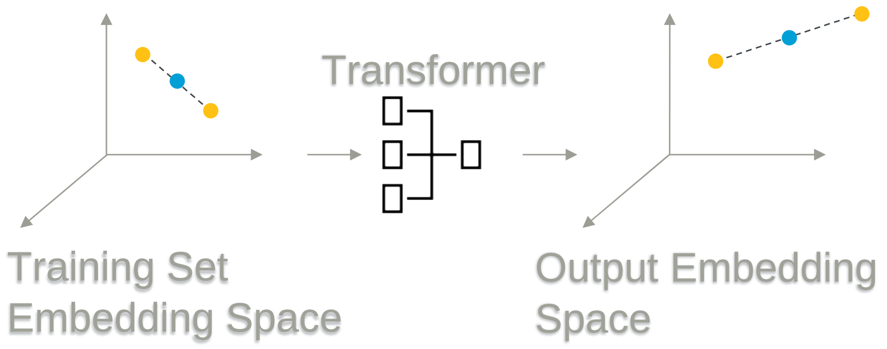

Figure 19 – C' the transformation of C keeping its spatial location related to A' and B'

To interpolate embeddings, we first had to sort training set labels into couples. Our first attempt was coupling each even label number with its following odd number (0-1, 2-3, 4-5, …). For each couple we interpolated their samples linearly and stopped once one of the labels ran out of samples. For example, if our couple is made from a 3 samples label and a 5 samples label, the new interpolates label will contain 3 samples.
Since the number of samples for each label ranges from 1 to hundreds we received unbalanced couples and could only produce 10% of interpolated embeddings compared to existing embeddings going over the entire training set.

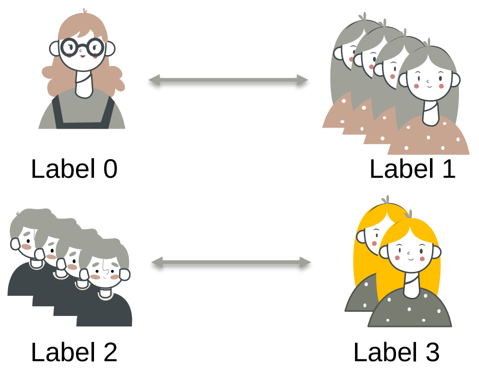

Figure 20 - Linear interpolation

Since we wanted to produce a bigger percentage of interpolated embeddings, we changed the sorting method to be according to the number of samples per label. That way we won’t face unbalanced coupling. We’ve also capped the number of samples in each new label by 30 and placed a minimum of 20. With this method we could produce the numbers we desired. In our training we used up to 50% if interpolated training embeddings.

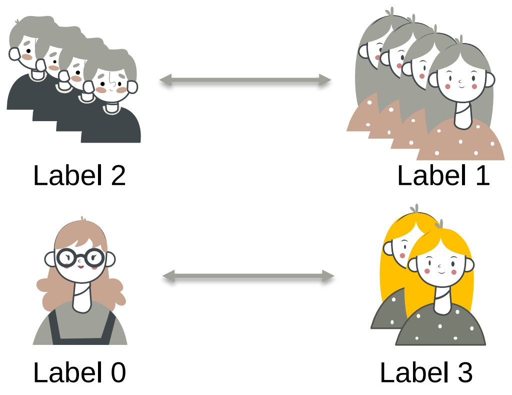

Figure 21 - By sample size interpolation

Evaluation Process

After training our Transformer we had to decide on an accuracy evaluation metric. We’ve decided on the Nearest Neighbor approach.

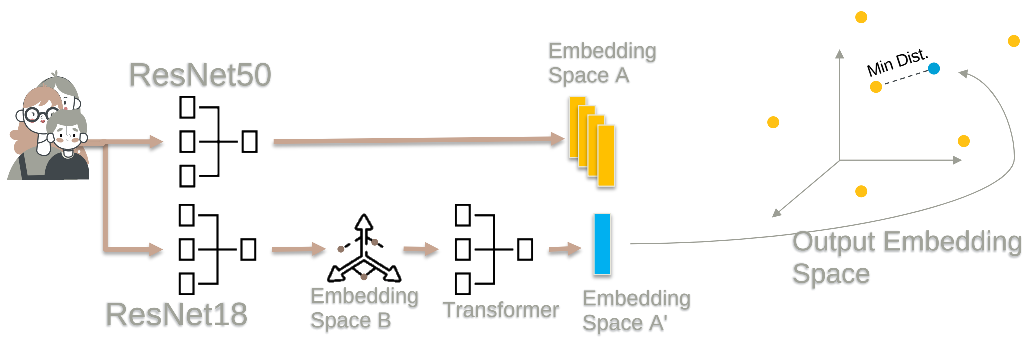

Figure 22 - Calculation accuracy using Nearest Neighbor

At first, we've decided to use a mean representation of the gallery space: We took the Gallery space and created a new one by representing each label by the mean of all its samples. The idea behind this is that we expect all samples of a label to be locally close in the Gallery space. By taking their mean we receive an approximate location of the label in space.
Then with the new mean Gallery space we can search for the closest mean representation to the transformed embeddings Euclidean-wise.
However, we've observed poor results. Using this method with the Gallery itself yields only 50% accuracy rate.

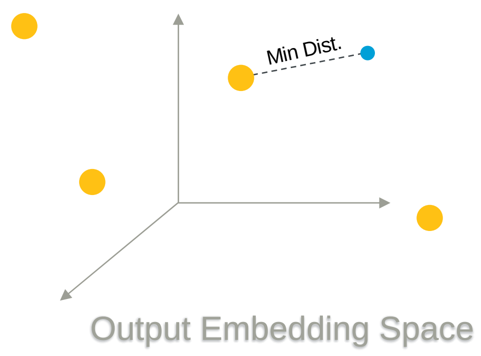

Figure 23 - Nearest Neighbor using mean of labels

We hypothesized the samples are not dense enough spatially.

Next, we used Nearest Neighbors on the Gallery without calculating the mean. As a consequence, we had to drastically reduce the size of the validation and test set since Nearest Neighbor is too time consuming, However, we received good and accurate results.

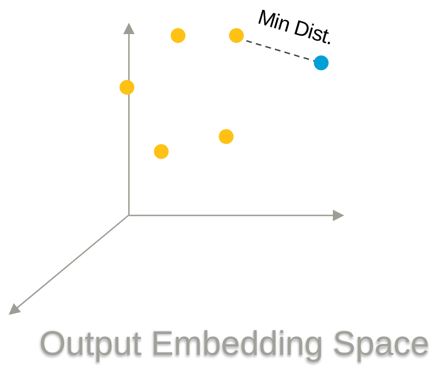

Figure 24 - Nearest Neighbor using unaltered Gallery

Training Process

Before doing actual training on the dataset we've used an Identity network for a sanity check to see how the Transformer will perform without any training and observed 0.2% accuracy rate.

We trained the MLP model using only the training set with a result of 77% accuracy. We then trained the same model with the training set combined with an interpolation rate of 50% and received results of around 74% accuracy. A decrease of 3%.

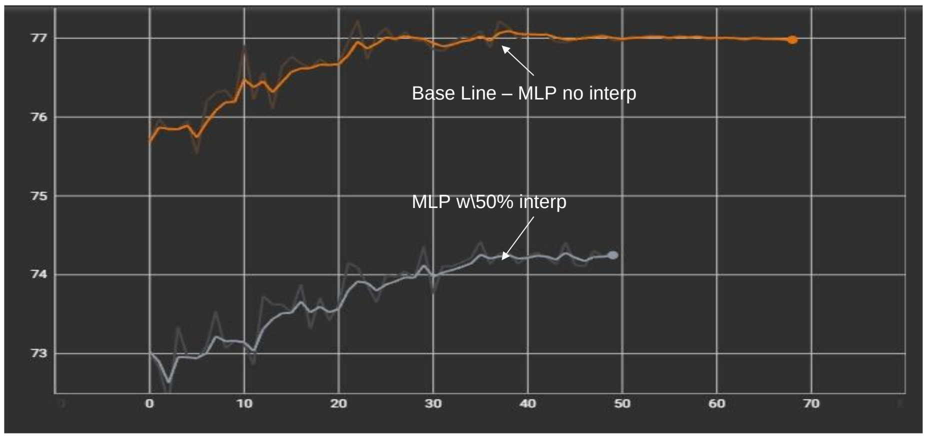

Figure 25 - Accuracy rate of MLP with and without 50% interpolation rate

We speculated the poor results stem from incorporating too many interpolated embeddings in the training process and decided to reduce their percentage to 25%. This time we received the same accuracy rate as without interpolation – 77%

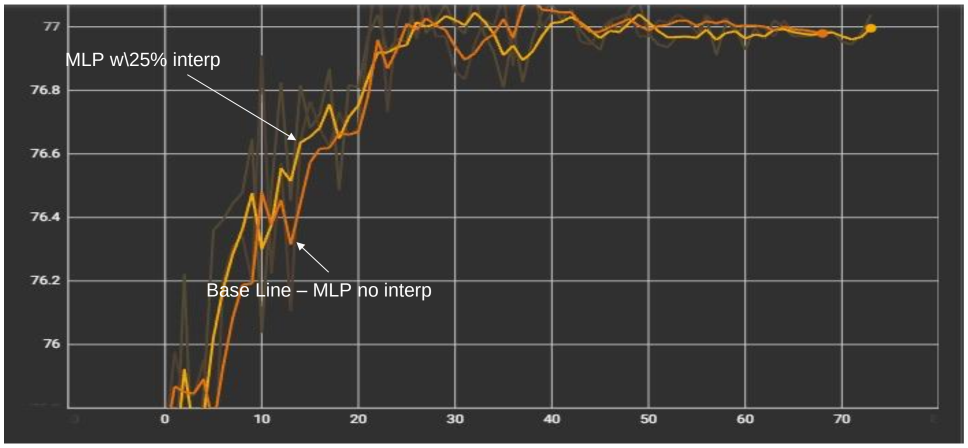

Figure 26 - Accuracy rate of MLP with and without 25% interpolation rate

Next, we've decided to split the training into two parts. The first is a pre-train where we train the model with the train set and 25% interpolation rate and the second is continuing training the model with the train set only. Here we've observed an accuracy of around 79% which is an increase of around 2%.

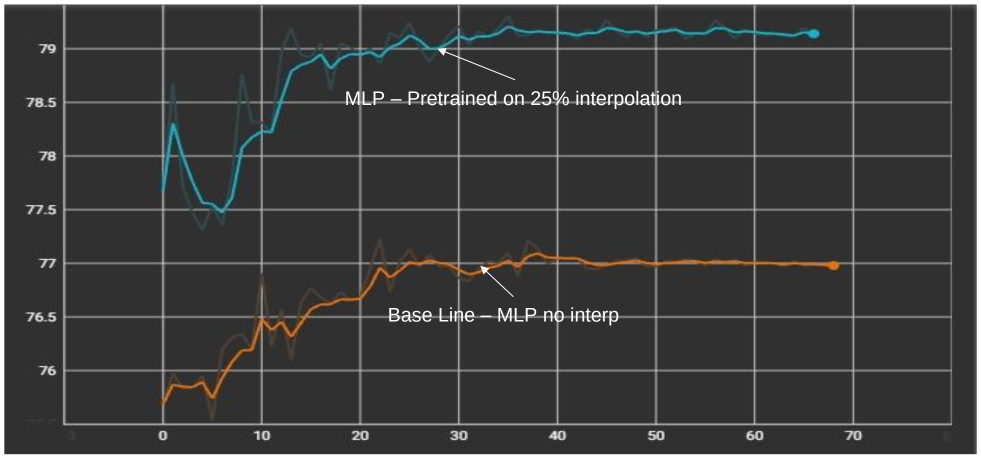

Figure 27 - Accuracy rate of pre-trained MLP with 25% interpolation rate

Results

Below is a table summary of the accuracy of different models with variation in interpolation method, interpolation rate and training method.
First we have the Identity network which had poor results as expected.
Then we have the Multi-Scale ResNet results which compared lower to MLP and so we dropped it after a couple of training sessions.
Next we have the MLP training sessions with linear and sorted interpolation methods, 0%/12%/25%/50% interpolation rate and regular and pretrained training.

Baseline result is bolded in yellow: Configuration is MLP without interpolation and has an accuracy rate of 76.98%

Best result is bolded in blue: Configuration is MLP, pretrained on 25% sorted interpolation rate and has an accuracy rate of 79.14%

Architecture	**Interpolation Method**	**Interpolation Rate**	**Training Method**	**Accuracy**
Identity	-	-	-	0.2%
MsResNet	-	-	-	61.31%
	Linearly	12%	Full Data	59.97%
MLP	-	-	Full Data	**76.98%**
	Linearly	12%	Full Data	60.43%
	Sorted	50%	Full Data	74.25%
			Pre-Train + Full Data	74.29%
		25%	Full Data	77.1%
			Pre-Train + Full Data	**79.14%**

Improvements

Dataset Infrastructure

In order to comply with our memory limitations, we used the PICKLE compression module for python.
It caused some logistic issues such as having to save the dataset as multiple different files and having to create a complicated dataloader which can deal with the data's fractured nature and support batch work while still being transparent to the user.
Improving this infrastructure by finding an alternate and friendlier dataframe than PICKLE can save a lot of time and ease use.

More Architectures

In this project we have experimented with MLP and MSResNet models. We dropped the MSResNet after observing subpar results, however that does not mean there are no better models to choose. Whether they are deep neural networks or even more classic algorithms.

Interpolation Method

Theoretically all Embedding Vectors in each space are placed on a specific high-dimensional topological surface inside the space called the Manifold. And in the same way each point on the Manifold theoretically corresponds to a possible Embedding Vector of a real image. We might suggest that our Transformer is learning the transformation between the Manifold of the Query and the Manifold of the Gallery.
Now considering the interpolated embedding, if it is on the Manifold then it does produce meaningful information for our Transformer to learn. It might as well have been another image in our dataset. However, if it is far from the Manifold, our Transformer can learn a wrong transformation and its accuracy will reduce.
The interpolation methods we've used are very naïve. They both sort the dataset into couples based on number of samples or just arbitrarily without considering their spatial location in space. If we couple two labels with samples far away from each other, we increase the chances of the interpolated embedding being far from the Manifold. Therefore, sorting the couples for interpolation based on proximity in space might increase the contribution of the interpolated data to the training process.

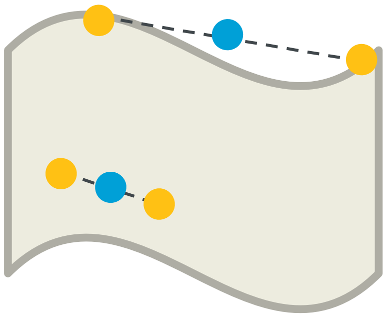

Figure 28 - Interpolated embedding out of bounds of Manifold

Embedding Space Proximity

We've observed using Nearest Neighbor with the mean of the Gallery labels did not yield good results, this led us to believe the samples of each label are not spaced close enough. Ideally, we would like them to be so because theoretically the deep features of a specific person should be embedded to the same general location on the Manifold. Perhaps with incorporation of regularizations or other methods we can achieve better results in this manner.

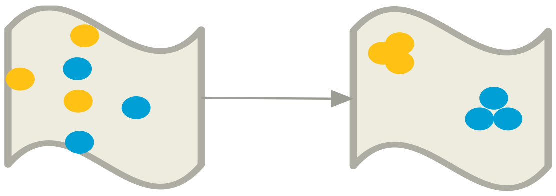

Figure 29 - Sparse spatial location of labels on the left to densely spaced on the right

Conclusions

Balancing

We've seen it is not enough to just throw in various loss functions and expect an improvement. We must also balance their contribution in order to enjoy from their strengths without them overshadowing each other.

Datasets are Tricky

Building a working and easy to use dataset is a difficult task that forces an attention to many details such as memory limitations and enabling a functioning API for a smooth user experience.

Deeper is Not Always Better

In out experiments we have seen how the shallow, one-layer MLP outperforms the deep multi-scaled ResNet. Deep neural networks are not always the better solution, and we should always question their specific usefulness in each task we are facing and whether we would benefit more from alternative, perhaps more classic solutions.

Synthetic Data Work When Used Correctly

In this project we observed how synthetic data can both hinder our results and improve them. It all depended on how and when we used it as well as how much. And of course, how well it was produced.

We should be cautious whenever we want to incorporate synthetic data and first understand very well the topology involved.