# Neural Network project, paper:RELATIVE REPRESENTATIONS ENABLE ZERO-SHOT LATENT SPACE COMMUNICATION

**Antonino Prinzivalli**
2058753
prinzivalli.2058753@studenti.uniroma1.it

## 1 Introduction

In this report, I will explain how I re-implemented from scratch several sections of the paper. Specifically, I focused on:
WORD EMBEDDINGS: In this section, I compare the embeddings after the relative transformation.
LATENT DISTANCE AS A PERFORMANCE PROXY: Here, I use CNNs to study the similarity of the representations with respect to performance.
ZERO-SHOT MODEL STITCHING: In this part, I stitch an AE and a VAE without additional training.
TEXT CLASSIFICATION: Finally, I perform text classification using the same classifier for two languages, training it on only one language.

## 2 AE Stitching

For this model i considered the MNIST dataset. The first step, before computing the relative space, is to choose anchors. Here, I select one anchor image from each class in the dataset. These anchors are then used in the encoder to help transform the input images into a latent space that is relative to these anchor images. The key part of this encoder is the relative projection function, which projects the input images into a latent space relative to the anchor images. I define two Autoencoders (AEs), train, and evaluate them separately, using the same anchor images for both. Afterwards, I define a class to perform the stitching by taking the encoder of the first AE and the decoder of the second AE. Finally, I evaluate the reconstructions without retraining, implementing zero-shot stitching.

## 3 relative VAE

For this model, I used the FashionMNIST dataset. The selection of anchors is similar to previous models(AE), but what is different here is the relative projection function, which projects the latent vectors z into a space relative to the anchor images. The decoder then reconstructs the input from these projected latent vectors. This is a generative model with a loss function that combines the reconstruction loss (mean squared error) and the Kullback-Leibler divergence. The training loop feeds each image batch into both VAEs separately over the specified number of epochs, computing and backpropagating their respective losses. I then create the stitched VAE by combining the encoder of VAE1 and the decoder of VAE2, loading their respective state dictionaries. At the end, I evaluate both the stitched relative VAE and the individual VAEs by analyzing their reconstructions with respect to the original images.

## 4 WORD EMBEDDINGS

Here, I use two models: the FastText and Word2Vec models. I started by finding the words that both models have in common, particularly focusing on the common vocabulary and applying simple filtering to eliminate words with little meaning. For visualization purposes, I selected four words with distinct meanings and their neighbors, forming four clusters to observe if the embeddings of these clusters are aligned in both models. Initially, I created the clusters using the FastText model, but I then filtered these clusters to ensure the words are valid in both models. From the resulting list of words, I selected 300 anchors to maintain the embedding dimensions of the models. I computed the absolute embeddings to compare them with the relative ones, and also the anchor embeddings, which are fixed and used for calculating the relative embeddings through the relative projection function. In the end, in the plotting functions, I computed the relative embeddings using the anchor embeddings of each model and the embeddings for each model of all the words in the clusters. This allowed me to visualize and compare the representations of these clusters in both models.

## 5 similarity of the latent spaces with CNN

In this part of the project, I aimed to implement the concept from the paper titled "4.2 LATENT DISTANCE AS A PERFORMANCE PROXY." However, since we didn't study Graph Neural Networks (GNNs) in this course, I substituted them with Convolutional Neural Networks (CNNs). The focus here is not on the CNN model itself but on demonstrating that models with similar performance can share similar embeddings. For time and computational reasons, I used four models. I defined a method to extract embeddings from the penultimate layer, which can be used to obtain feature representations of the input data. I used the CIFAR-100 dataset to ensure the task was sufficiently challenging, allowing for more variation in the representations. I trained models for different numbers of epochs (30, 40, 100, and 120). After training each model, I evaluated the test loss and accuracy. Finally, I computed the cosine similarity between the embeddings of the models to analyze the relationship between performance(more epochs means better performance in this example) and embeddings.

## 6 TEXT CLASSIFICATION

This part of the project involves loading pre-trained models and tokenizers for English and Italian. The RobertaModel and RobertaTokenizer are used for English, while AutoModel and AutoTokenizer are used for Italian. DATASET: The dataset used in the paper, which involved Amazon reviews, was no longer accessible. Therefore, I opted for the tyqiangz multilingual-sentiments dataset from Hugging Face. This dataset contains multilingual sentences, making it suitable for my goal of training in English and evaluating in Italian, and it is structured as a classification task. To find the anchors, we use sentences since we are working with models that create contextual embeddings. I selected a number of sentences in English and, instead of using Google Translate as done in the paper, I used another model to obtain their Italian translations(commented part in the code). This translation process consumes a lot of RAM, so I saved the translated anchor sentences in a file (translations.txt) to avoid repeating the translation from scratch. To run the code, you should pass the translations.txt file in Colab. After setting up the anchors, I trained the classifier with embeddings produced by Roberta, initially without any relative transformation. After training and evaluating the classifier with the absolute embeddings, I trained it using the relative transformation and evaluated it both in English and, finally, in Italian.