Optimizing Intent Classification with BERT

Jack Kai Lim
Halıcıoğlu Data Science Institute
University of California, San Diego
jklim@ucsd.edu

Vivian Chen
Halıcıoğlu Data Science Institute
University of California, San Diego
vnchen@ucsd.edu

Hou Wan
Halıcıoğlu Data Science Institute
University of California, San Diego
hwan@ucsd.edu

Elsie Wang
Halıcıoğlu Data Science Institute
University of California, San Diego
e2wang@ucsd.edu

Abstract

Recently, transformers have achieved remarkable performance in many different tasks. In this paper, we use a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model for user intent classification. For training and evaluation, we utilize the Amazon MASSIVE dataset, which encompasses diverse languages and intents. Our approach involves fine-tuning the BERT model, followed by the incorporation of advanced training methods, and employing contrastive learning to assess model performance. The results reveal significant improvements with the application of advanced techniques. With the baseline model, before and after fine-tuning the BERT model, we achieve a modest accuracy of around 1%. However, by incorporating the first advanced technique of warm-up steps, we see a substantial increase in test set accuracy to 86.55%. The combination of two techniques further enhances this to 86.75%. Notably, contrastive learning methods such as SupCon and SimCLR yield impressive accuracies of 81.41% and 87.73% respectively, demonstrating their effectiveness in improving model performance. This demonstrates the robustness and versatility of our approach in enhancing user intent classification and also contributing valuable insights into the application of advanced training techniques and contrastive learning for further advancements in natural language understanding.

1 Introduction

Recent advancements in natural language processing have witnessed the remarkable success of transformer-based models, achieving promising performance across a diverse range of tasks. In this paper, our focus is on using a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model for the purpose of classifying user intent. Our methodology starts with the fine-tuning of the BERT model to establish a solid baseline. Next, we employ a range of training techniques derived from insights found in a blog to construct a customized model tailored to our specific task. Finally, our investigation will extend to training models using contrastive losses, delving into advanced strategies for enhancing model performance.

1.1 BERT

For our paper, we fine-tuned a basic BERT model, a bidirectional transformer designed to pre-train deep bi-directional representations from a combination of masked language modeling objective and next sentence prediction on a large corpus comprising the Toronto Book Corpus and Wikipedia

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

[Devlin et al., 2018]. The model variation we used is the English uncased model which changes input text to lowercase and strips out accent markers. The model is primarily aimed to be fine-tuned on texts on tasks such as token classification, making decisions, or question answering. This makes the model suitable classifying user intent.

1.2 Dataset

The dataset used to train our model is the Amazon MASSIVE intent dataset[FitzGerald et al., 2022]. It consists of 1 million examples of realistic, human-created virtual assistant utterance text spanning 51 languages, 60 intents, 55 slot types, and 18 domains. For classification tasks (except for 3.3 Contrastive Learning), the input to the model is the text and the output is the user intent label. For instance, with the input text "wake me up at nine am on friday", the user intent label of this text is 48 (alarm_set).

The paper that introduced this dataset FitzGerald et al. [2022] highlighted the need for the need for evaluation datasets for specific tasks to keep up with the growing number and size of large language models like BERT. Additionally, despite the rise and advancement of virtual assistants, they only support a small number of the world's 7,000+ languages. Thus, the MASSIVE dataset was created with the aim of providing labeled data for training and evaluation, particularly data that is realistic for the task and is natural for each given language. With this paper, we hope to utilize the MASSIVE dataset to advance the linguistic analysis of intent classification on BERT and provide insight into improving virtual assistants by using a larger and newer dataset and employing various methods.

2 Related Work

Many papers have cited this dataset for fine-tuning and evaluation performance. For instance, one paper used the dataset to fine-tune a data generation method called Language model INstruction tuning to Generate annotated Utterances for Intent classification and Slot tagging (LINGUIST) [Rosenbaum et al., 2022]. LINGUIST generates annotated data for Intent Classification and Slot Tagging by fine-tuning AlexaTM 5b, a 5-billion-parameter multilingual sequence-to-sequence (seq2seq2) model. The MASSIVE dataset, given its large range of languages and intents/slots, was used as training to demonstrate the cross-domain adaption capabilities of LINGUIST, including cross-lingual and cross-schema transfer. Training on this dataset improved the ST F1 score, seeing a particularly large improvement for Japanese.

Another paper used a subset of this dataset in Russian to evaluate model performance on multilingual BERT models [Karpov and Burtsev, 2023]. The article explores the effectiveness of knowledge transfer from the RuQTopics dataset, a comprehensive Russian topical dataset. Russian-only models trained on this dataset consistently achieved an accuracy of around 85% on a subset matching the Russian MASSIVE classes. The findings revealed a strong correlation between model accuracy and the approximate size of BERT's pretraining data for each language, emphasizing the significance of training sample size in multilingual models.

To fine-tune our model, we took inspiration from the Medium blog post, "Advanced Techniques for Fine-tuning Transformers", which proposes advanced training techniques for fine-tuning Transformers such as BERT [Chang, 2021]. The author used the same model and dataset and compared the mean RMSE of many techniques: layer-wise learning rate decay (LLRD), warm-up steps to a learning rate schedule, re-initializing pre-trained layers, Stochastic Weight Averaging (SWA), and frequent evaluation. The author found improvements in results for all techniques, with a combination of all techniques performing the best. For our paper, we used a combination of two techniques, warm-up steps and re-initializing pre-trained layers.

For Contrastive Learning, we took inspiration from the Supervised Contrastive Learning paper's [Khosla et al., 2020] contents for our training pipeline that uses contrastive learning on sentences for the intent classification. The article itself explores extening self-supervised batch contrastive learning to a fully-supervised setting, effectively utilizing label information. The supervised contrastive (SupCon) loss was used, achieving a top-1 accuracy of 81.4% on ImageNet dataset using ResNet-200 and outperforming cross-entropy on various datasets. We also looked at SimCSE paper [Gao et al., 2021] and used their findings of augmentation via hidden dropout to generate similar pairs of inputs for natural language data. The article introduced SimCSE, a contrastive learning framework that

predicts the input sentence istelf in a contrastive object, utilizing standard dropout as minimal data augmentation. It incorporated annotated pairs, and evaluation on semantic textual similarity tasks using BERTbase showed an average Spearman's correlation of 76.3% and 81/6%, demonstrating its effectiveness.

3 Methods

3.1 Baseline Model

For our baseline model, we used a basic pre-trained BERT architecture to train our data. The hidden layer dimension, set at 10, corresponds to the number of attention heads in multi-head attention mechanisms within BERT. The embedding dimension, fixed at 768, represents the size of hidden states and output of attention layers. Training was conducted for a single epoch. Furthermore, we used a batch size of 16, a learning rate of 1e-4, and a dropout rate of 0.1. For our optimizer, the Adam optimizer was utilized for its adaptive learning rate capabilities, which help in converging faster and more effectively compared to standard Stochastic Gradient Descent. For loss, we used cross entropy due to its effectiveness in classification tasks, including classifying user intent, where the task can be considered classification over the vocabulary.

3.2 Custom Fine-tuning Model

For our custom fine-tuned model, we implemented a linear scheduler with applied warm-up steps and re-initialized pre-trained layers. Instead of using the StepLR scheduler from the baseline model, we use a learning rate scheduler that uses the 'get_linear_schedule_with_warmup' function with 50 warm-up steps. This type of scheduler is used when training neural networks with a warm-up phase, which allows the model to start with a low learning rate and eventually increase it during the initial training steps. The linear schedule means the learning rate increases linearly during the warm-up phase and decreases linearly during the later remaining training steps. We also re-initialized 4 pre-trained layers, and this discards some of the weights from the model's pre-trained weights and re-initializes them during the training process, which is supposed to cause better fine-tuning results. Other than these two changes, we mostly used the same hyperparameters and settings as the baseline model.

3.3 Contrastive Models

For our contrastive models, we used the SupCon loss from the SupCon loss paper's [Khosla et al., 2020] repository. The normal training of the classifier is preceded by a contrastive training loop which aims to take two different augmentations of the same input and pushes similar inputs together, and dissimilar inputs away in embedding space. Two models are trained, one with SupCon which is a supervised method of training, and the other with SimCLR which is an unsupervised method. Once this is done, the layers used for contrastive training are frozen, then a linear classifier is trained for downstream classification. We opted to omit a scheduler from the contrastive training steps due to achieving worse performance with it, and used the AdamW optimizer to train the model. The downstream classification uses similar methodology to our custom fine-tuning model, employing a StepLR scheduler with warmup steps.

4 Results

4.1 Test Loss and Accuracies

4.2 Train and Validation Accuracy

All training and validation accuracy will be taken from the 10th epoch for each training example.

exp idx	exp	loss	accuracy
1	Test set Before Fine Tuning	765.9042	0.0098
2	Test set After Fine Tuning	755.6769	0.0148
3	Test set with 1^{st} technique	176.5817	0.8655
4	Test set with 2^{nd} technique	757.8145	0.0138
5	Test set with 2 Techniques	192.5612	0.8675
6	Test Set with SupContrast	32.2872	0.8141
7	Test Set with SlimCLR	9.9618	0.8773

Table 1: Table for test set experiments

Model	Exp	Train Accuracy	Validation Accuracy
Baseline	Before Fine Tuning	0.0127	0.0103
Baseline	After Fine Tuning	0.0107	0.0128
Advanced Techniques Model	with 1^{st} technique	0.9944	0.9966
Advanced Techniques Model	with 2^{nd} technique	0.0144	0.0103
Advanced Techniques Model	with 2 Techniques	0.9806	0.9784
SupConstrast	N/A	0.8173	0.9611
SlimCLR	N/A	0.9865	0.9051

Table 2: Table for test set experiments

5 Discussion

5.1 Question 1

Q1: If we do not fine-tune the model, what is your expected test accuracy? And why?

A1: We expected a low test accuracy because BERT is trained on a large dataset of unpublished books, meant for basic English understanding. BERT is meant to be fine-tuned for downstreamed tasks. Thus, without fine-tuning, BERT would not perform well with specific tasks such as accurately labelling user intent. Furthermore, an optimizer is not used, which prevents the process of adjusting the weights and biases of a neural network to minimize the loss function.

5.2 Question 2

Q2: Do results match your expectation(1 sentence)? Why or why not?

A2: These results do not match with our expectations, as we expected the model to improve a lot then just 0.5%. I think the reason for that is because we only tried fine tuning the model by adjusting the learning rates and using the schedulers/loss functions we deemed appropriate from previous models such as the FCN CNN etc.

5.3 Question 3

Q3: What could you do to further improve the performance ?

A3: We were thinking of possibly more advanced techniques that are more related to the intent model we are trying to create, and also use custom tuning techniques that are more beneficial and related to improve the Intent Model

5.4 Question 4

Q4: Which techniques did you choose and why?

A4: Warm-up Steps and Re-initializing Pre-trained Layers. We made our scheduler a layer-wise learning rate decay with applied warm-up steps because it helps stabilize the training process during the initial phase. Sudden large updates to the weights during training from scratch may lead to instability, so slowly increasing the learning rate helps the model avoid large weight updates in the early steps. Furthermore, this method also allows helps the model initialize better. Since as described before, warm-up steps help the model explore the parameter space more carefully in the beginning

due to the fact the weights do not increase in large amounts. As the model trains more, the learning rate eventually increases, which allows the model to make larger updates.

We also chose to reinitialize some of the earlier layers from the Bert Model. This is because the Bert Model was trained to perform Mask Language Modeling (MLM) and Next Sentence Prediction (NSP) however our model that we are trying to create is an intent model, which does have similarities in what we are trying to accomplish, but there is a sizable difference in terms of the output we are trying to achieve. Hence, we reinitialize some layers to give it a clean slate in learning our task, while still maintaining all the encoding and the more general and useful information from the bottom levels, we reinitialize n of the layers closer to the output in order to fine tune and tailor our model to our task better.

5.5 Question 5

Q5: What do you expect for the results of the individual technique vs. the two techniques combined?

A5: We expected the 2 techniques combine to perform a lot better then with just one of the individual techniques. As in a way, both techniques help to improve the already trained model, by improving the models ability to determine intent based of the input we give it.

5.6 Question 6

Q6: Do results match your expectation(1 sentence)? Why or why not?

A6: The results did not match our expectations. As incorporating the second technique gave us a negligible improvement compared to the fine tune model. In addition, the model did not perform any better between incorporating the first technique versus having both techniques implemented.

Some reasons, we think that is was the result are firstly, initially we thought that the output pattern/type of the model although similar would have a sizable enough difference to where re-initializing the last couple of layers would improve the models performance to perform the intent task. However, due to the results that we got, there is reasonable evidence to suggest to us that the outputs are similar enough to the point where there is close to no benefit to re-initializing the layers.

However, if we are just looking at the first technique we used which is the warm-up steps, we can see a huge improvement between the fine tune model and the model that incorporates this technique which is what we expected.

5.7 Question 7

Q7: What could you do to further improve the performance?

A7: Things we could further improve for the model are to maybe add additional layers on top of the Bert model. We could also try out other special loss functions that could be better suited for the intent task. Furthermore, we could also attempt the other techniques mentioned in the article, such as adding a stochastic weight averaging scheduler and frequent evaluation.

5.8 Question 8

Q8: Compare the SimCLR with SupContrast. What are the similarities and differences?

A8: The similarity between the two is that both use a contrasting loss function to pull positive pairs close together and push negative pairs apart in embedding space, which helps learn useful representations. The difference is that SimCLR is self-supervised while SupContrast is supervised. SimCLR does not use label information; it instead uses two views of the same image as the positive pair and the other images in the batch as negatives. On the other hand, SupContrast does use label information, so it able to sample many positives per image anchor from the same class in the batch while it treats images of different classes as negatives.

5.9 Question 9

Q9: How does SimCSE apply dropout to achieve data augmentation for NLP tasks?

A9: First, the input sentence is fed to the encoder twice with an independently sampled dropout mask each time, which constructs two versions of the embedding for the same sentence. Then, these two embedding are treated as a positive pair, for they are a representation of the same underlying sentence while the differently dropped out words are treated as a minimal form of data augmentation between the positive pair. Next, the other sentences within the mini-batch are treated as negative instances. The contrastive loss mentioned in the last question is then used to pull positive pairs close together and push negative pairs apart in embedding space, which helps learn useful representations. Lastly, standard dropout noise causes the creation of slightly different views of the sentences, so this makes enough signal for contrastive learning.

5.10 Question 10

Q10: Do the results match your expectation? Why or why not?

A10: Overall, our SimCSE results perform relatively well with SupCon having an accuracy of 0.81 and SimCLR having an accuracy of 0.88. The accuracy of 0.88 is the highest performing model out of all models, and is expectedly so with the upstream pretraining that brings similar inputs together in embedding space.

However, our results do not meet our expectations in the sense that a supervised method should expectantly perform better than an unsupervised method due to having more access to more information. After many iterations and trials of different hyperparameters such as changes in learning rate, schedulers and temperatures, we were not able to reliably get SupCon to perform better than SimCLR. This could perhaps be because of not being able to determine the right hyperparameters, or even could be due to the quality and relevance of data representation; SimCLR, by maximizing agreement between different augmented views of the same sentence, may capture a broader and more nuanced semantic understanding that aligns well with the specifics of intent classification. This unsupervised approach is not affected by labeling noise, which can adversely affect SupCon's performance by introducing inaccuracies into the supervised learning process.

5.11 Question 11

Q11: What could you do to further improve the performance ?

A11: To boost model performance, we could explore more on hyperparameter experimentation, such as adjusting learning rates, augmentation techniques, and loss function settings. Additionally, perhaps implementing a hybrid approach where a model is first trained on SimCLR for unsupervised pre-training to capture broad features from unlabeled data, and then refined with features with SupCon through supervised fine-tuning on labeled data could be a viable strategy to improve robustness and performance. This leverages the strengths of both unsupervised and supervised learning, potentially leading to more effective and nuanced model performance.

6 Authors' Contributions

6.1 Vivian

Pair programmed with Jack on the Baseline and Custom models. Assisted in getting rid of the numerous indentation errors in the initial code. Wrote the about the Custom Fine-tuning Model in the Methods section, recorded some of losses and accuracies. Also answered half of question 4, generated some of the plots, and wrote the readme file.

6.2 Hou

Pair programmed with Elsie on the Contrastive learning models. Wrote the answers for all questions regarding the results and implementation of the contrastive models, and ran both models to get accuracies as well as loss plots. Wrote part of Related Work and Abstract.

6.3 Elsie

Pair programmed with Hou on Contrastive learning models and ran them. For the report, wrote Discussion Q1, the Introduction, most of Related Work, part of Abstract, and description of the Method baseline model.

6.4 Jack

Pair programed with Vivian on the Baseline and Custom models. Implemented the plot function. Assisted with debugging for the supcon model. Also wrote the answers for questions 3, 4, 5 and part of 6 and 7. Also ran some of the models to get the train loss and accuracy and the validation accuracy.

References

Peggy Chang. Advanced techniques for fine-tuning transformers, Nov 2021. URL `https://towardsdatascience.com/advanced-techniques-for-fine-tuning-transformers-82e4e61e16e`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, et al. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *arXiv preprint arXiv:2204.08582*, 2022.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

Dmitry Karpov and Mikhail Burtsev. Monolingual and cross-lingual knowledge transfer for topic classification. *arXiv preprint arXiv:2306.07797*, 2023.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

Andy Rosenbaum, Saleh Soltan, Wael Hamza, Yannick Versley, and Markus Boese. Linguist: Language model instruction tuning to generate annotated utterances for intent classification and slot tagging. *arXiv preprint arXiv:2209.09900*, 2022.

A Plots

A.1 Baseline with Fine Tuning Accuracy Plot

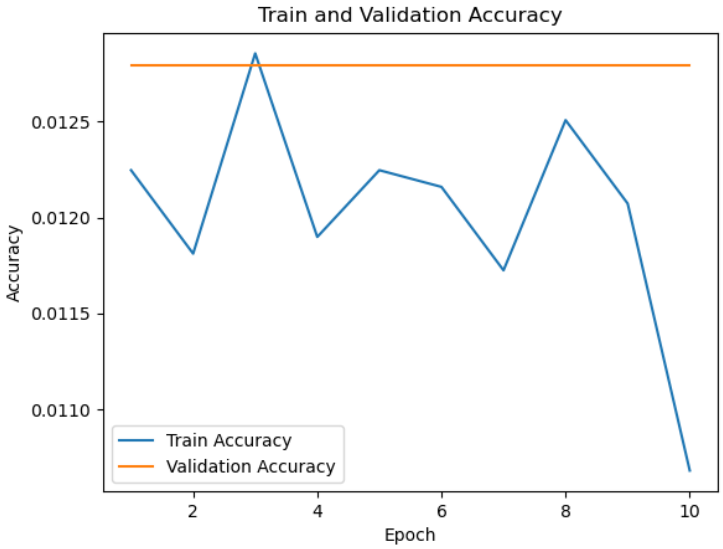

Figure 1: Baseline with Fine Tuning Accuracy

A.2 With 1st technique Warm Up Steps

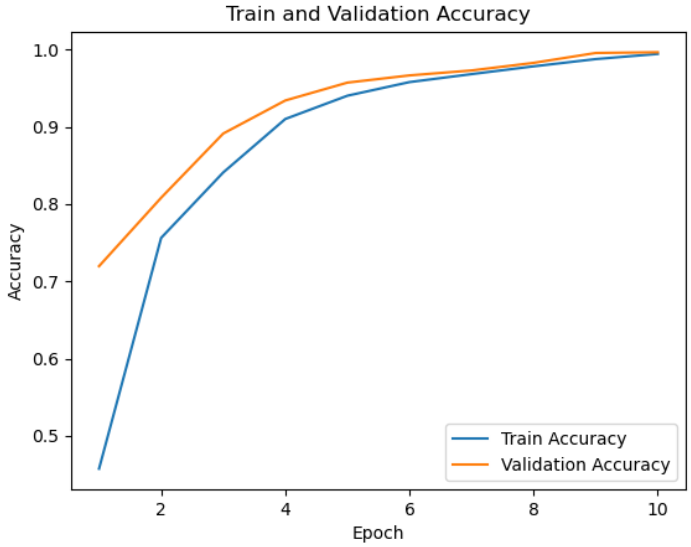

Figure 2: With Warm Up Steps Only

A.3 With 2nd technique Reinit Last Layers

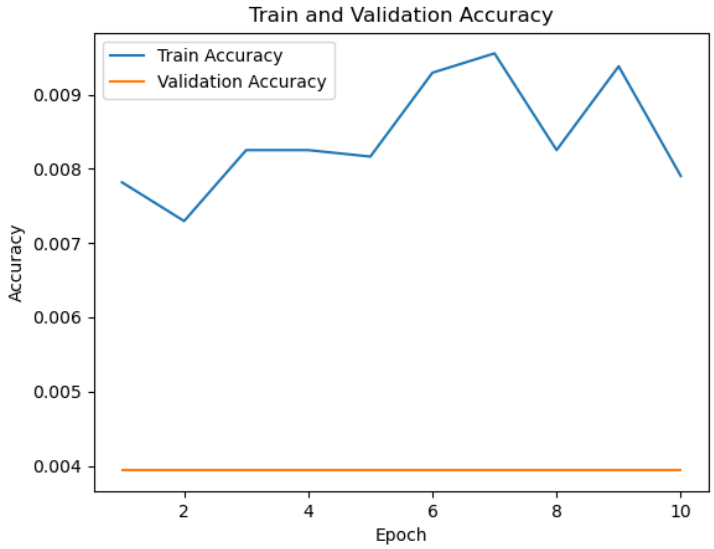

Figure 3: With Reinit Last Layers Only

A.4 With SupCon loss contrastive learning

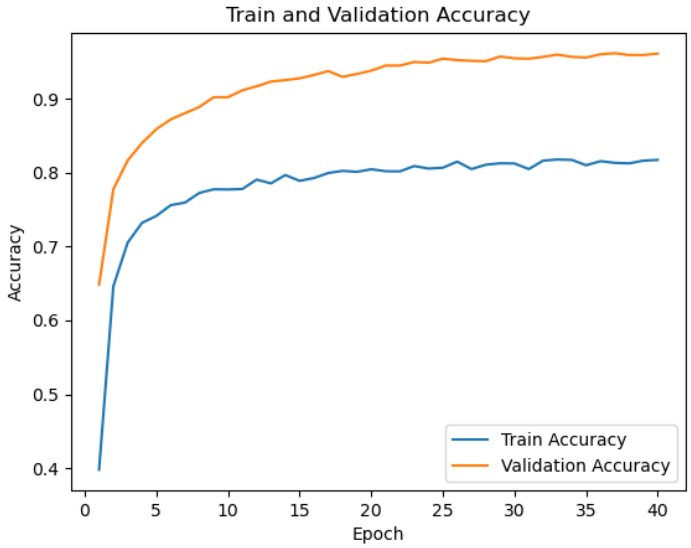

Figure 4: Supervised contrastive learning

A.5 With SimCLR loss contrastive learning

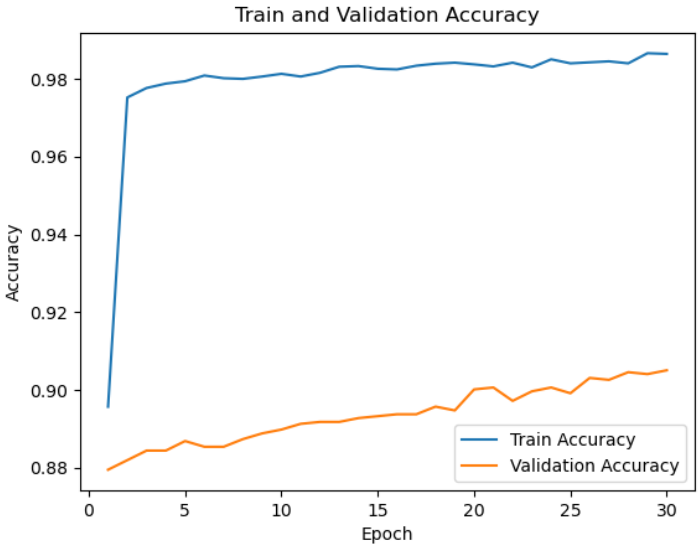

Figure 5: Unsupervised contrastive learning