**Project Title**
Studying the Performance Of Self-Supervised Models

**Team Members**
Vishwesh Kirthivasan (vk2470) and Hitesh Agarwal (HA2598)

**Abstract**
Self-supervised learning methods are extremely popular now, owing to the fact that there is a lot of unlabelled data (crawling text from various articles, images from the web, etc) and because annotating large amounts of data is very costly. Self-supervised learning eliminates the need to annotate all of the data available, instead, learning the distribution of the data in a hierarchical way, first learn using a self-supervised way using an autoencoder for example. This uses unlabelled data. Then using the embeddings from this "pre-trained" network, we can fine-tune a much smaller amount of labeled data to classify the inputs into distinct classes. But how does the performance of the overall classification task improve by keeping the amount of labeled data available fixed, while increasing the amount of unlabelled data for the pre-trained model? What is the tradeoff between time to accuracy and wall clock time when we increase the amount of data? We will be using MNIST and CIFAR-10 for this analysis.

**Motivation**
We find that the most prevalent and closest relationship to our task is language models in the field of NLP, where gigantic base models (BERT, GPT, ELMo, etc) are trained in an unsupervised fashion with just textual data and no labels. The advent of these models really brought a paradigm shift to NLP. Exploring such models is already extremely hot in the field of Computer Vision.

**Why not Existing Base Nets?**
Quite often, base nets are trained on a completely unrelated task, with unrelated data. So the features are not very easily transferable.

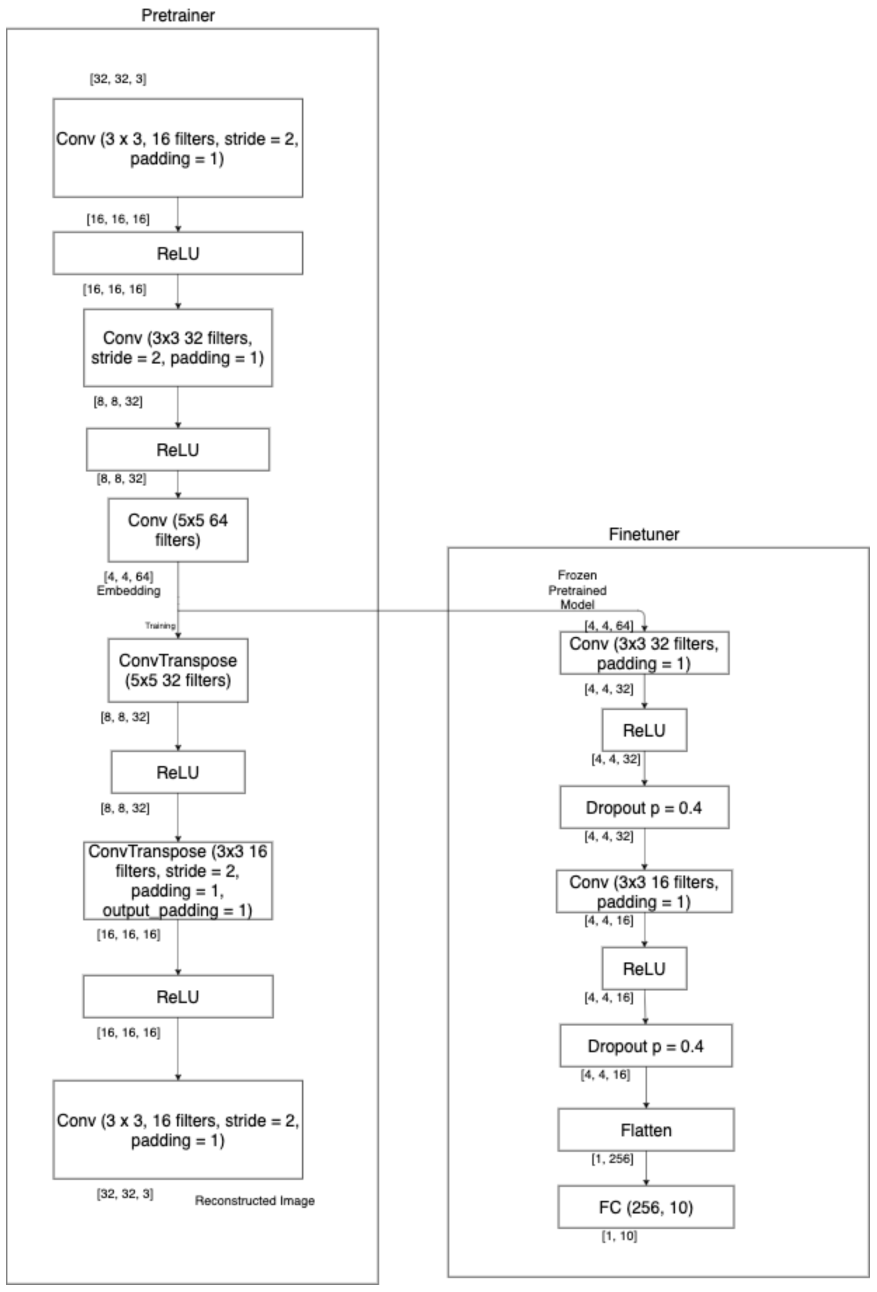
**Relevance and Challenges**
This project is extremely relevant to the current research areas in the field of Deep Learning, owing to the amount of cost that can be saved by reducing the amount of labeled data. The biggest challenge that we faced was in getting the model to generalize well, especially when we pass in less unlabelled data to the autoencoder, and less labeled data to the fine-tuning network. We solved this by data augmentation (Gaussian noise, vertical flip, and horizontal flip) and adding dropout. Further, we faced the problem of pretrainer completely getting overridden during fine-tuning if we trained both networks (even if we had a less learning rate for the base net). We hence froze the base net. It also made sense in this context, because we want to analyze the behavior

with increasing unlabeled data. To override the model that uses unlabeled data defeats that purpose here.

## Dataset Used

We have used CIFAR-10 as our dataset for this project.

## Model



1. The model will be split into two parts:

    a. Pre-trained model - that takes unlabelled data and learns and embedding space to represent the input data. We used an autoencoder for this purpose. An autoencoder is an encoder-decoder architecture that aims to find an optimal representation of the input data, and then reconstruct the input itself from the learned representation. We used an RMSE loss to compare the reconstructed output and original input.
    b. Fine-tuned model - that takes labeled data and learns how to classify the above embeddings of inputs into distinct classes. We used a combination of convolutional networks and one fully connected layer as shown in the flow chart above.

**Technique**

1. For a particular amount of labelled data and unlabelled data:
    a. Train the pretrainer in the following way:
        i. Use the unlabelled data, pass through the autoencoder
        ii. Compare the reconstructed output of the autoencoder and the input using RMSE
        iii. Calculate gradients and update weights
    b. Train the finetuner in the following way:
        i. Use the labeled data, pass through the encoder part of the autoencoder.
        ii. Pass the resultant learned representation as input to the finetuner
        iii. Compare the predicted classes with the ground truth classes for the image using cross-entropy loss.
        iv. Calculate gradients and update weights
2. Repeat for different amounts of labelled and unlabelled
3. Plot the graphs and discuss inference. The graphs that would be plotted are:
    a. Accuracy metric vs Epochs (Time to accuracy)
    b. Wall clock time for each iteration
    c. Amount of data (labelled and unlabelled) needed for a particular accuracy
4. For each value of the amount of labeled data, also train a model that trains the encoder from the autoencoder and the finetuner in tandem (this is our baseline, completely supervised).
5. Compare the plots for self-supervised with the completely supervised network.

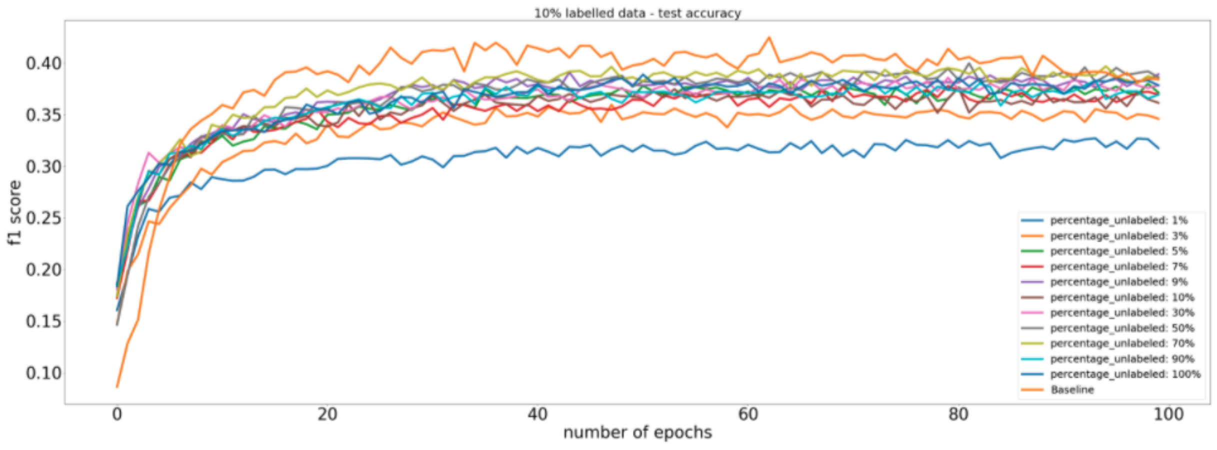
**Implementation Details**

1. GPU - NVIDIA Tesla V100
2. Framework - PyTorch
3. Dataset - CIFAR-10

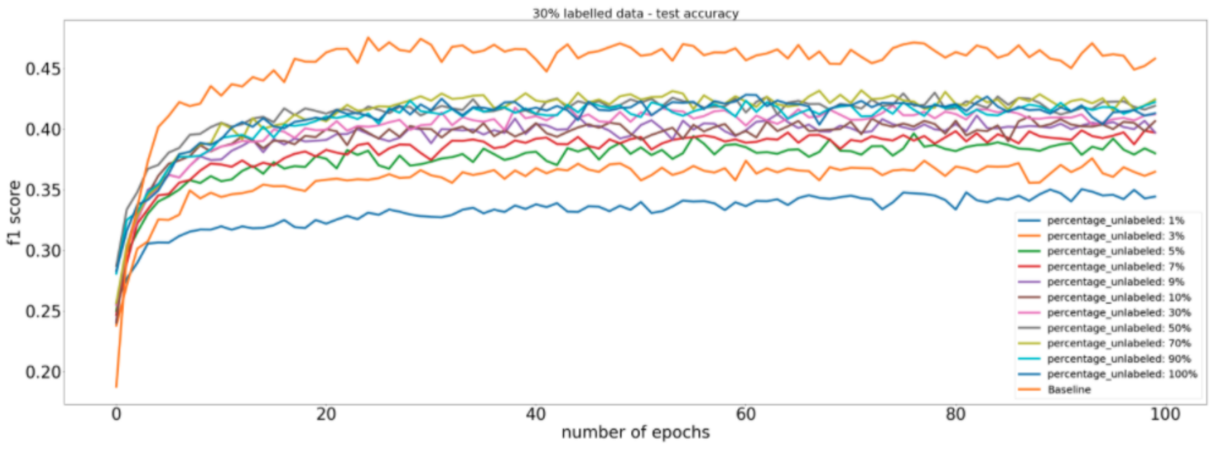
## Results

Visualizations of the various comparisons mentioned above and deriving insights from the same.

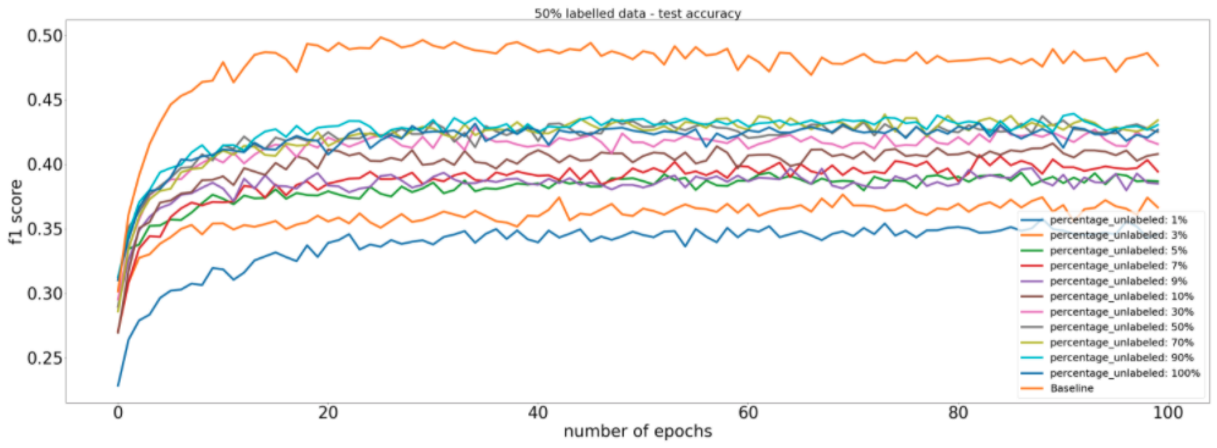
### F1-Score - 10% Labeled Data



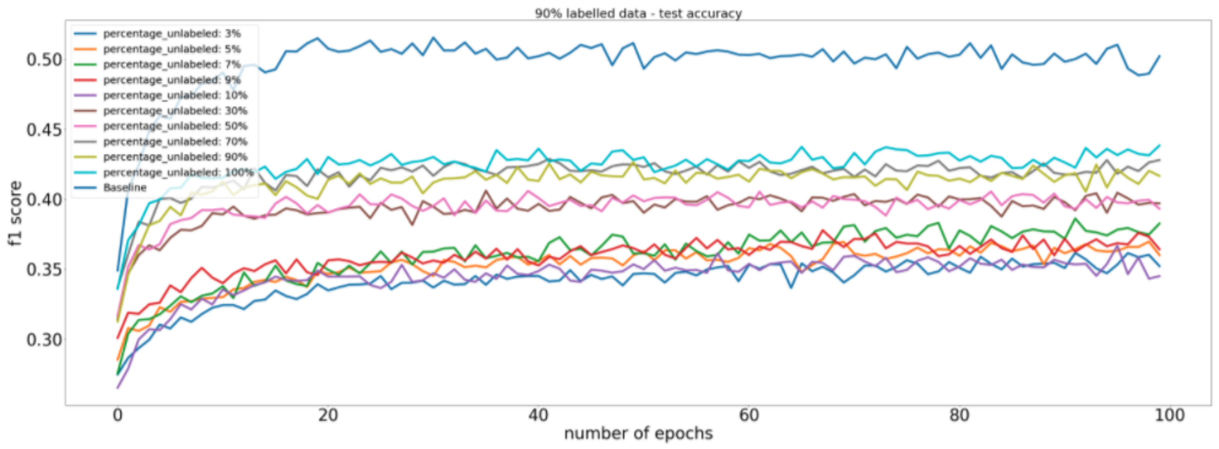
### F1-Score - 30% Labeled Data



### F1-Score - 50% Labeled Data



### F1-Score - 90% Labeled Data



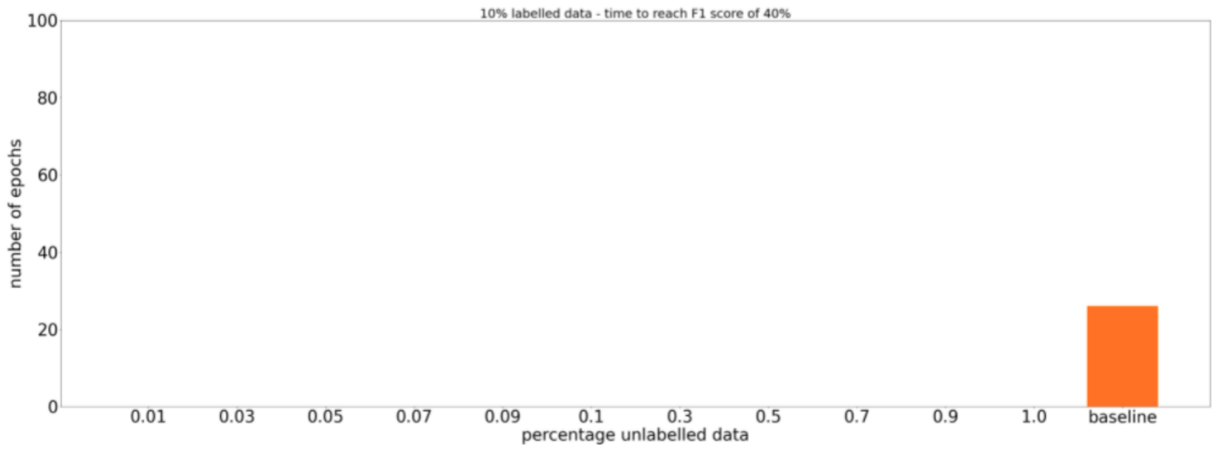
The key takeaways from the F1-Score graphs are:

1. Increase in scores as amount of unlabelled data increases
2. Increase in differentiability in the scores when labeled data is increased

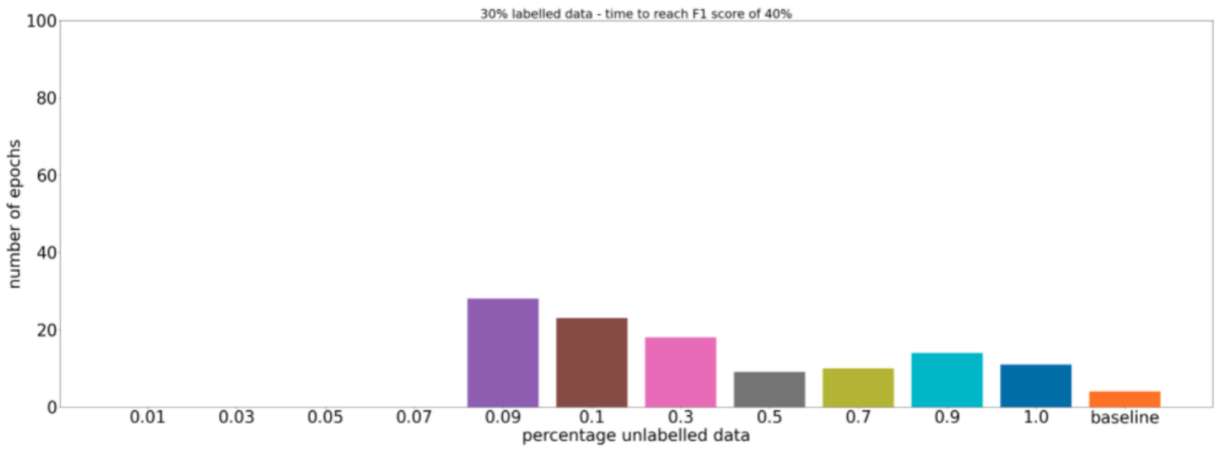
### Time to Accuracy Graphs

If a particular model did not reach 40%, we won't have a bar graph corresponding to that model.

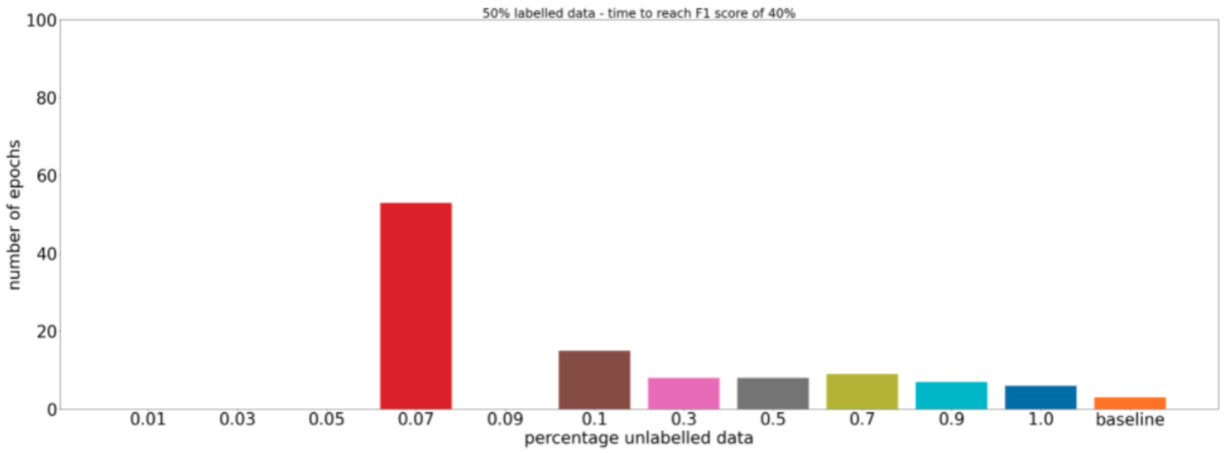
### Time to Accuracy - 10% Labeled Data



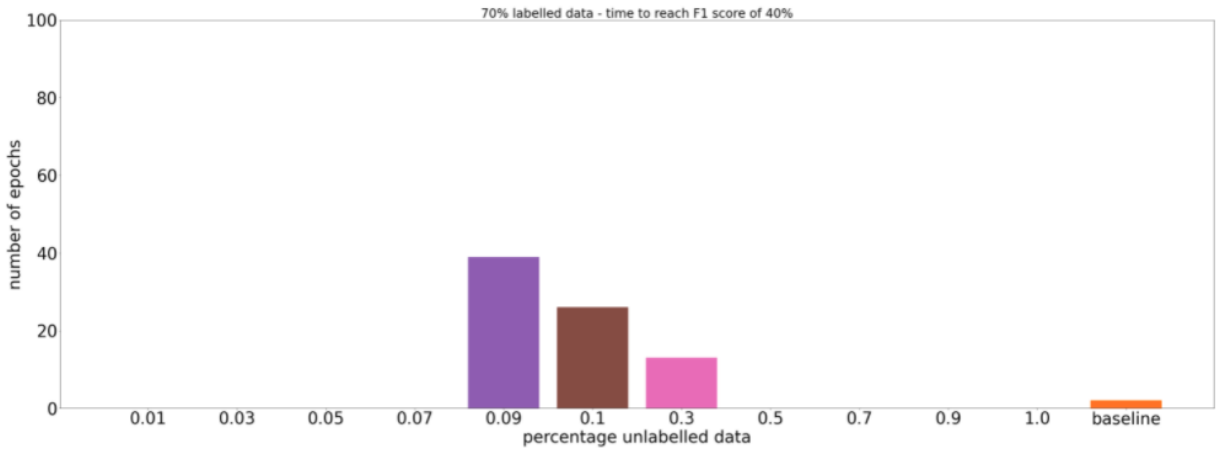
### Time to Accuracy - 30% Labeled Data



## Time to Accuracy - 50% Labeled Data



## Time to Accuracy - 70% Labeled Data



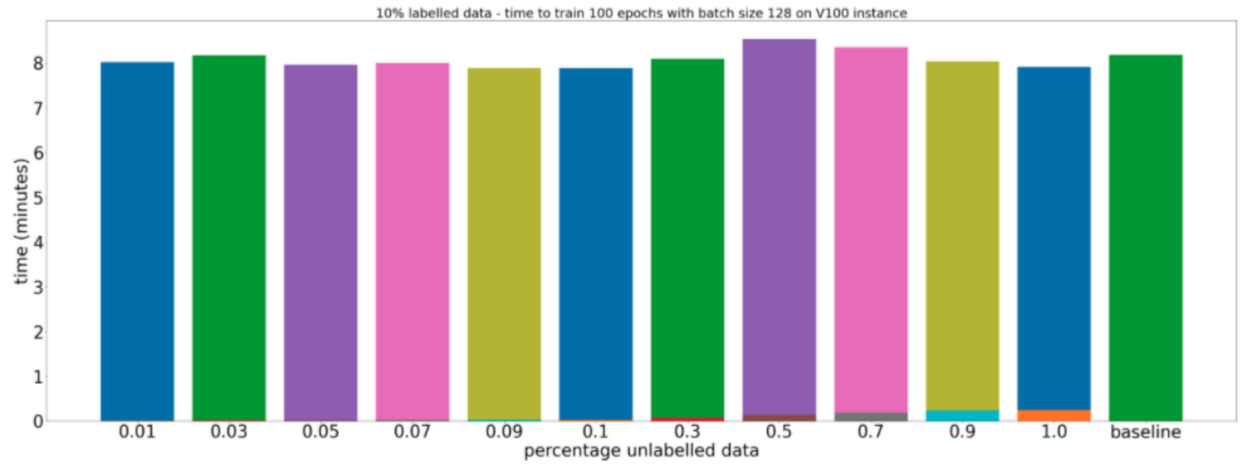
The key takeaways from time to accuracy graphs:

1. Decrease in time to accuracy with increase in percentage unlabelled data
2. Decrease in time to accuracy with increase in percentage labelled data
3. Increase in time to accuracy after a particular level of unlabelled data (maybe because of overfitting in the pretrainer).

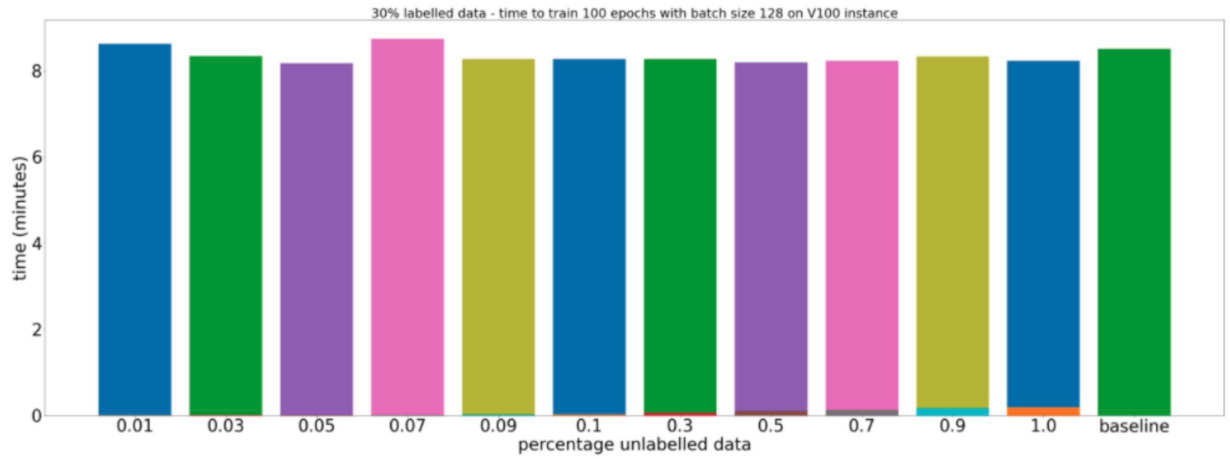
## Wall Clock Time

We ran our experiments in a machine with V100 GPU, with a batch size of 128.

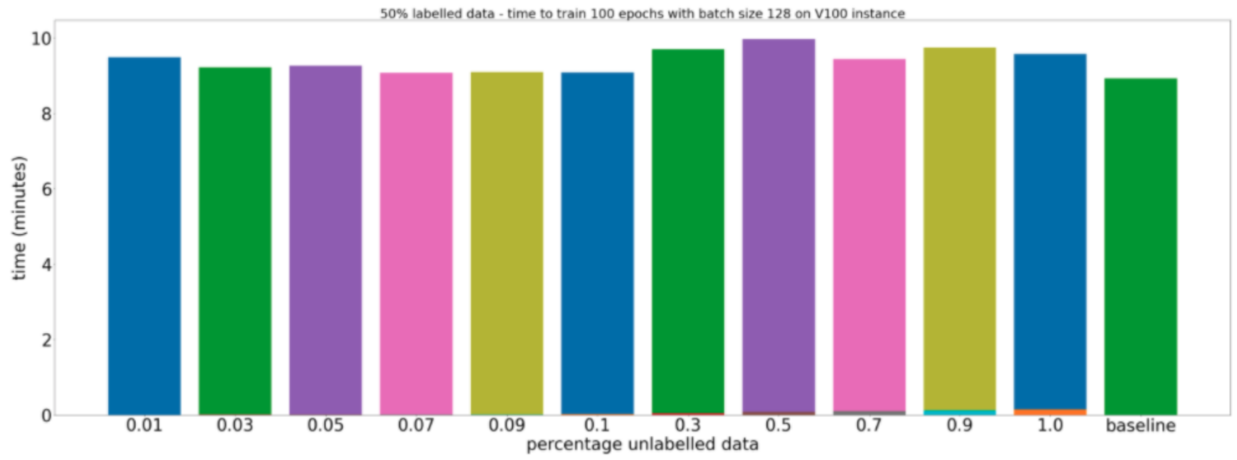
## Wall Clock Time - 10 % Labeled Data



## Wall Clock Time - 30% Labeled Data



## Wall Clock Time - 50% Labeled Data



The fact that there is not much of a difference between the different wall clock times is a good sign, because experimentation will be more deterministic and fast.

**Inference**

1. Firstly, we can observe that there is an increase in performance in the classification task if we increase the amount of data to the pretraining model.
2. We noticed that increasing the amount of unlabeled data does not always mean that the performance improves. Specifically, what we noticed was that there is a point at which the performance plateaus, and increasing the amount of unlabelled data at that point does not guarantee an increase in performance (but it will not degrade also). This will obviously be different for different values of labelled data. In our case, this seemed to be at around 10% of unlabelled data when we used 50% of the labelled data, for example. The explanation for this is that the distribution of the data, for the capacity of the model, is captured by all but 10% of the total available data.
   This has an important application in practice because, for a fixed amount of labelled data, we can easily get more unlabelled data. So, when we find that the performance plateaus after some level of unlabelled, it is an indication that we may need to focus on the model complexity before collecting more data.

**Future Work**

1. Use more powerful models
2. Test on multiple datasets
3. Use other methods for pretraining
   a. Contrastive Loss
   b. GANs

**References:**

Papers (some of these papers were not directly used in implementing, but more for the purposes of learning for bettering our model's performance):

β-VAE: https://openreview.net/pdf?id=Sy2fzU9gl
Understanding the Behaviour of Contrastive Loss: https://arxiv.org/pdf/2012.09740.pdf
Unsupervised Learning Using Generative Adversarial Training and Clustering: https://openreview.net/pdf/16c049b9a075af86f550b81707b62d5e520daed1.pdf
Transfusion: Understanding Transfer Learning for Medical Imaging: https://arxiv.org/pdf/1902.07208.pdf

Blogs:

Self Supervised Learning
https://towardsdatascience.com/self-supervised-learning-methods-for-computer-vision-c25ec10a91bd
Contrastive Loss:
https://towardsdatascience.com/contrastive-loss-explaned-159f2d4a87ec