# A2 Research

## Semi-Supervised Learning

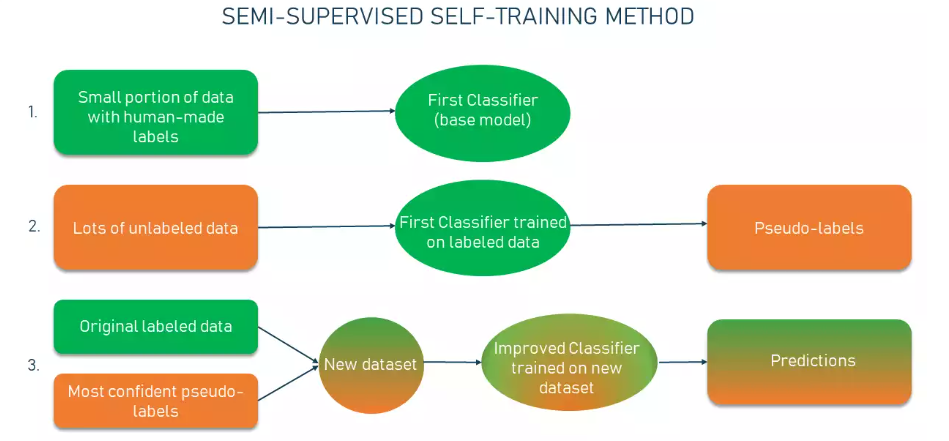
Also called Weak Learning

<https://www.altexsoft.com/blog/semi-supervised-learning/>

Technique to train using both labelled data and also unlabelled data.

The basic premise is this:

1. Train a base model on the labelled data
2. Apply **pseudo-labelling:** 
   1. This is where you predict on some or all of the unlabelled data.
   2. Then, you try to rate the confidence of your predictions.
   3. Find the most confident of the predictions, for example greater than 80% confidence
   4. Take these most confident predictions and combine them with the original labelled dataset.
   5. Then, train a second model, based on both the labelled and most confident predicted data
   6. This process can iterate (10 rounds is common)



There is another similar process called **Co-training.** Haven’t read yet into the details but apparently it’s about training two individual classifiers based on two views of the data

# Research Findings and Report Guide

* Evaluation Metrics
  + IsCancerous, use F1 Score
  + CellType, use Accuracy
* Basic Decision Tree model was generated in file 04, but results were not good. Best not to consider as a Base line Model
* PyTorch Baseline Model 2 Layer Fully Connected Neural Network in File 05c.
  + No Image Preprocessing to grey scale has been done
  + Results: IsCancerous
    - Training Accuracy: 0.961
    - Training F1 Score 0.945
    - Test Accuracy: 0.873
    - **Test F1 Score: 0.894**
  + Results: CellType
    - Training Accuracy: 0.867
    - **Test Accuracy: 0.79**
* Tensorflow Base model was created, in Assignment2 notebook.
  + Image Preprocessing applied, converted to greyscale
  + 2 Layers
  + Results: IsCancerous
    - Training Accuracy 89%
    - Test Accuracy: 82%
  + Results: CellType
    - Training Accuracy: 76%
    - **Test Accuracy: 70%**
* PyTorch 3 Layer Fully Connected Neural Network in File 06.
  + No Image Preprocessing to grey scale has been done
  + Results: IsCancerous
    - Training Accuracy: 0.955
    - Training F1 Score 0.936
    - Test Accuracy: 0.863
    - **Test F1 Score: 0.884**
  + Results: CellType
    - Training Accuracy: 0.883
    - **Test Accuracy: 0.784**
* Experimented with More layers in files 07 and 08
  + No significant improvements on Accuracy for Cancerous or F1 for Cell Type, but slower training time
  + To Rerun on Colab (don’t have results saved, haven’t updated for test, to put in 06 file)
* Tuned Learning Rate in file 09
  + Picked 0.0001 as best learning rate
  + 0.00003 has slightly better results in experiments, but was very slow, not worth the time tradeoff
* **Analysis**: As can be seen from the FC NNs, There is a large gap between training results and test results
  + Expand on Bias and Variance
    - IsCancerous have relatively low training error, therefore low bias, high variance, overfitting
    - Cell Type have higher training error, possibly due to complexity in data/classification, possibly bias. Higher variance, overfitting
    - Fix with Regularization
* Applied Regularization in file 14
  + Redo for 2 layer models. Run Files 14 and 15, record results in file 14
  + Tried different weight decays, with worse results, so use 0.0001
  + Tried different Dropout rates with worse results, so use 0.5
  + Conclusions
    - L2 not very effective
    - Dropout effective in reducing overfitting
    - Early stopping not very effective, but also good at reducing training time
  + Dropout Results: IsCancerous
    - Training Accuracy: 0.925
    - Training F1 Score 0.899
    - Test Accuracy: 0.882
    - **Test F1 Score: 0.907**
  + Results: CellType
    - Training Accuracy: 0.803
    - **Test Accuracy: 0.801**
* **Analysis**: Variance and overfitting reduced. But Training time is high and accuracy in both can possibly be improved.
* Next, Convolutional Neural Networks tested. Results see File 22
  + Tried 3 Convolution layers with 2 Layer Classifier NN, 3 Convolution layers with 3 Layer Classifier NN, with and without Early Stopping
  + Chose 3 Convolution layers with 3 Layer Classifier NN With Early Stopping. Good Results, some without Early stopping had insignificantly higher metrics, but longer training times
  + Results: IsCancerous
    - Training Accuracy: 0.948
    - Training F1 Score 0.927
    - Test Accuracy: 0.876
    - **Test F1 Score 0.899**
  + Results: CellType
    - Training Accuracy: 0.81
    - **Test Accuracy: 0.796**
* **Analysis:** Models are an improvement. Less of a Gap in error, in this case overfitting seems like a lesser issue or not an issue. But still possibility for improvement. Try some of the following methods
  + More Complex CNN models – more complex convolutions and/or Classifiers
  + Additional data – Use Extra Labels data
  + Data Augmentation
* Tried More Complex CNNs. See Files 24 and 25, see 25 for results
  + First, Tried more complex Convolution Layers. Still only 3 Convolution Layers, but increased the size of feature maps (channels) and increased kernel size of first layer (give reasons)
  + Second, Tried more complex Classifier Layers. Increase to 4 layers and increased Neurons
  + More Complex Convolution Layers gave better results, but more complex classifiers did not. Therefore, did not try combination of both, as training time has already significantly increased.
  + Results: IsCancerous
    - Training Accuracy: 0.946
    - Training F1 Score 0.927
    - Test Accuracy: 0.886
    - **Test F1 Score 0.909**
  + Results: CellType
    - Training Accuracy: 0.837
    - **Test Accuracy: 0.793**
  + For Is Cancerous, the more convolutions gave better results
  + For Cell Type, a slight reduction in Accuracy, so therefore not worth using
* Using Full Data For the Cancerous Model. The Extra data is labelled for Cancerous, so just added the data and trained. Training process is very slow:
  + Results: IsCancerous
    - Training Accuracy: 0.956
    - Training F1 Score 0.926
    - Test Accuracy: 0.849
    - **Test F1 Score 0.851**
  + Even though we added more data, the results are actually worse than before. Overcomplicated data? Or Overfitting?
    - Try again with the simpler CNN model
    - Also try with adding Dropout