Detecting Targets in Radar Signals

Adrian Iordache

December 29, 2021

Abstract

In this project, we will present a solution based on Shifted Windows Vision Transformers (Swin Transformers) for detecting (counting) various objects in radar signals under interferences.

1 Introduction

The main idea of this project was a comparison between state of the art Convolutional Neural Networks and Vision Transformer Networks.

For state of the art Convolutional Neural Network, we will consider Efficient Net architecture, pretrained on noisy students. Those type of networks represent a scalable result of neural architecture search optimized for a multi-objective metric between FLOPS and accuracy, containing the core layer from the MobileNetV2, inverted residuals bottleneck layers.

On the other hand, Vision Transformer Networks represent a new powerful trend when it comes to Computer Vision. Originally from text, transformer networks faced challenges when adapting from the initial domain to vision. Those challenges arise from differences between those two domains (text and vision), especially when it comes to the scale of high resolution images compared to text.

For that reason Shifted Windows Vision Transformers, represent a variation that proposed a solution for constructing hierarchical feature maps by merging patches in the deeper layers of the network, using multiple transformer layers with limited attention.

For our experiments we will begin in the baseline stage with:

- Efficient Net B5 (Image Size: 456, Trained on Noisy Students)
- Swin Large Transformer (Image Size: 384, Pretrained on ImageNet)

2 Dataset Description

For this task we have available a trainset with 15500 samples of labeled with one of five classes from 1 to 5, uniformly distributed, representing the number of objects in that images. Each class from 2 to 5 will have 3000 examples and only the first class will contain 500 samples more than the others.

The evaluation phase of this project will be done based on the inference of to 5500 samples using as test metric the accuracy score.

3 Project Stages

For keeping track of the experiments, this project was organized in stages, based on the end of each stage we will be able to extract conclusions for experiments in the further stages.

Project Structure:

- Stage-0: Baseline Models (Various Architectures, Augmentations, Ideas)
- Stage-1: Grid Search over large hyper-parameter space for a certain architecture
- Stage-2: Adding Stochastic Weight Averaging (SWA) and reducing the hyper-parameter space to a certain region based on the conclusions from Stage-1
- Stage-3: Switching to 10 Folds Cross-Validation, Pseudo-Labeling and Final Models Ensembles

4 Baseline Models (Stage-0)

In this stage, we are interested to oscillate between architectures, ideas, augmentations just so we can get a better "intuition" about the context we are trying to model.

For this we will try to maintain as fixed as possible hyper-parameters like: batch-size, learning rate, loss function, optimizer, learning rate scheduling procedure. Those hyper-parameters will be optimized in the later stages of the project.

In Stage-0 during the phase of Error Analysis, there were discovered images with a strong level of noise, to better identify them it was trained a model (EfficientNet B0) to classify our dataset between noisy images and clear images.

Based on approximately 1200 selected images with and without noise the models for detecting noisy images achieved 97% mean accuracy over 5 folds of cross validation.

After inferring the noise detection models on the train and test dataset we discovered the following:

- No. of train noisy samples: 4278
- No. of train clear samples: 11222
- No. of test noisy samples: 1500
- No. of test clear samples: 4000

And with this new information we will be able to see the impact of the noisy data in our original problem.

During Stage-0 experiments:

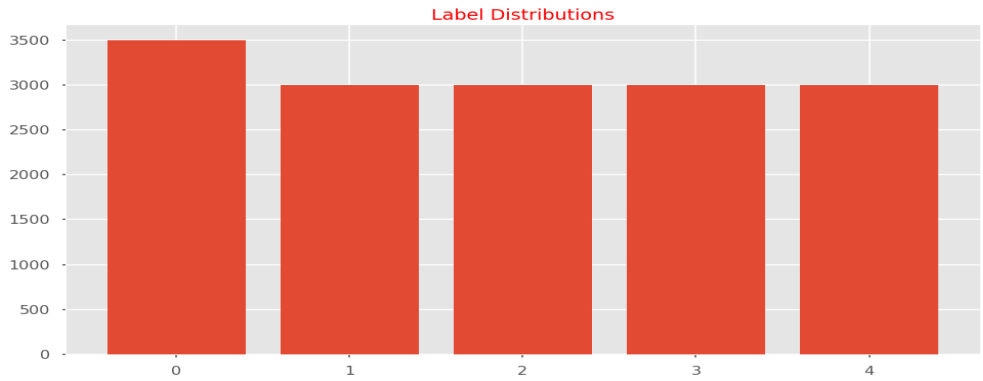
- We mainly fixed the following hyper-parameters:
  1. Learning Rate used 1e-5
  2. The learning rate scheduler used was Cosine Annealing Warm Restarts
  3. The loss function was Cross Entropy Loss
  4. And the optimizer used was AdamW
- We experimented with the following hyper-parameters:
  1. Backbone Architecture:
     (a) EfficientNet B5 (pretrained on Noisy Students)
     (b) Swin Large Transformer (Image Size: 384, Window Size: 12, Pre-trained on ImageNet)
     (c) BEiT Large Transformer (Image Size: 224, Pre-trained on ImageNet)
  2. Augmentations:
     (a) Spatial Level Transforms: (ex: ShiftRotateScale, Grid Distortion, Elastic Distortion, Horizontal Flip, ..)
     (b) Pixel Level Transforms: (ex: Random Contrast, Random Brightness, ..)
  3. Removing Noisy Samples
  4. Removing Samples predicted wrong by a large difference (more than 1)

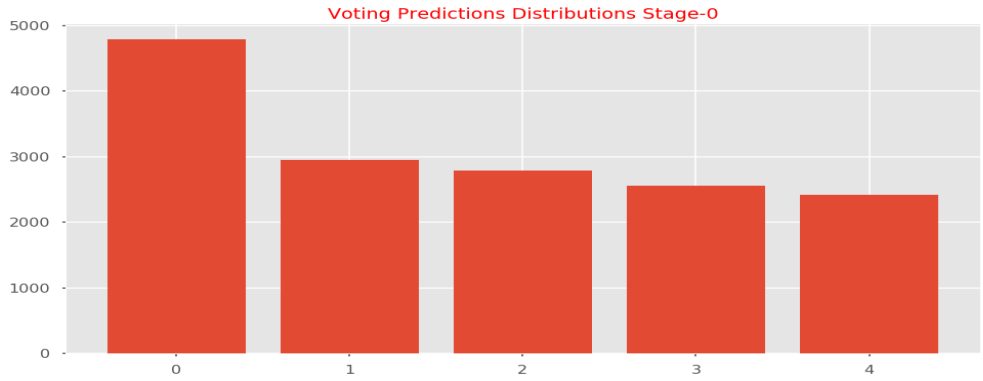
After the experiments in the Stage-0 we reached the following results:

Voting accuracy of all experiments: 0.747

For the following results we will consider when the voting ensemble is correct or wrong, when all models are wrong and when all models are wrong by a large margin (difference bigger than one).

Voting Wrong: 3911 (25.2%) $\longrightarrow$ Noisy: 1969 (50.3%) $\longrightarrow$ Clear: 1942 (49.7%)
Voting Good: 11589 (74.8%) $\longrightarrow$ Noisy: 2309 (19.9%) $\longrightarrow$ Clear: 9280 (80.1%)
All Wrong: 2194 (14.2%) $\longrightarrow$ Noisy: 1206 (55%) $\longrightarrow$ Clear: 988 (45%)
Very Wrong: 614 (4%) $\longrightarrow$ Noisy: 460 (74.9%) $\longrightarrow$ Clear: 154 (25.1%)





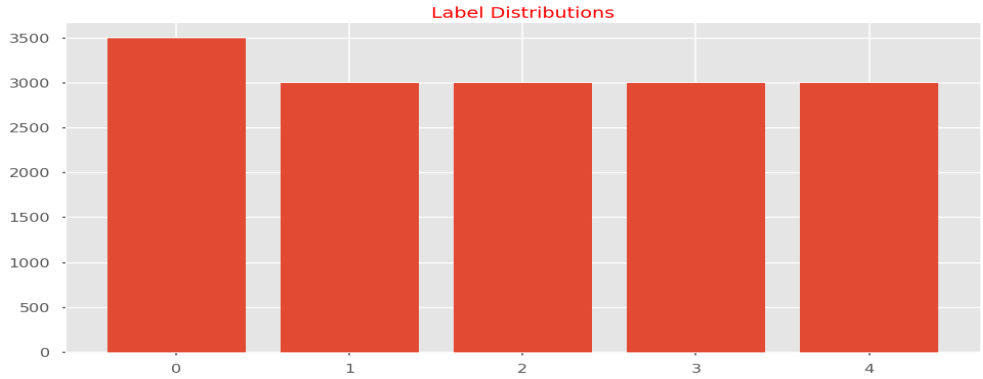
Stage-0 Conclusions: Based on the experiments in this stage we fixed the backbone architecture for the Shifted Window Vision Transformer and we established that high level of augmentations lowers the validation accuracy.

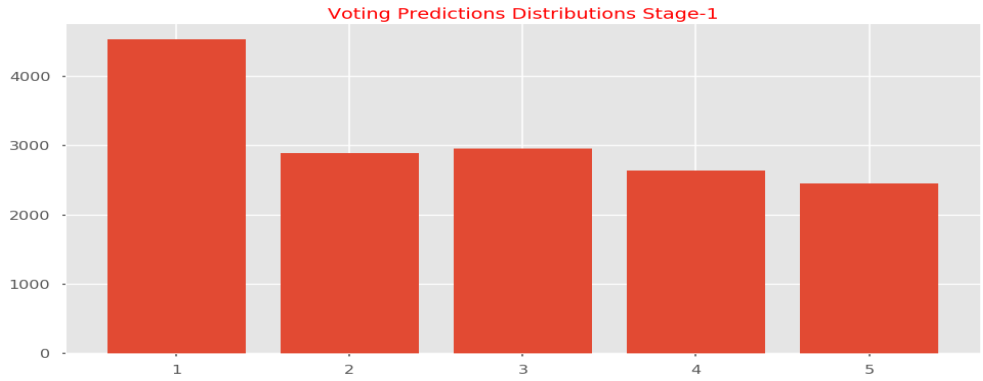
5 Grid Search (Stage-1)

During Stage-1 we used a grid search methodology to explore the hyper-parameters space. For those experiments we trained only one fold out of five, assuring the reproducibility of each experiment.

Experimented Hyper-parameters during Grid Search (Stage-1):

- Optimizers: AdamW, RangerLars
- Learning Rate: 7e-6, 1e-5, 3e-5, 5e-5, 7e-5, 1e-4, 3e-4
- Learning Rate Scheduling:
  1. CosineAnnealingWarmRestarts, T_0: 200, T_mult: 2
  2. CosineAnnealingWarmRestarts, T_0: 500, T_mult: 5
  3. OneCycleLR, max_lr: lr * 10
- Augmentations:
  1. None
  2. Horizontal Flip, Vertical Flip, Random Rotate 90, each with 50% probability of appliance





Selecting the best models from Stage-1 we achieved the following results:
Voting accuracy of experiments: 0.768, with best single model: 0.763
Voting Wrong: 3598 (23.2%) $\longrightarrow$ Noisy: 1849 (51.4%) $\longrightarrow$ Clear: 1749 (48.6%)
Voting Good: 11902 (76.8%) $\longrightarrow$ Noisy: 2429 (20.4%) $\longrightarrow$ Clear: 9473 (79.6%)
All Wrong: 2790 (18%) $\longrightarrow$ Noisy: 1480 (53%) $\longrightarrow$ Clear: 1310 (47%)
Very Wrong: 840 (5.4%) $\longrightarrow$ Noisy: 611 (72.7%) $\longrightarrow$ Clear: 229 (27.3%)
Stage-1 Conclusions: From those experiments we conclude on going further with AdamW as the optimizer and no augmentations.

6 SWA and Specialized Grid Search (Stage-2)

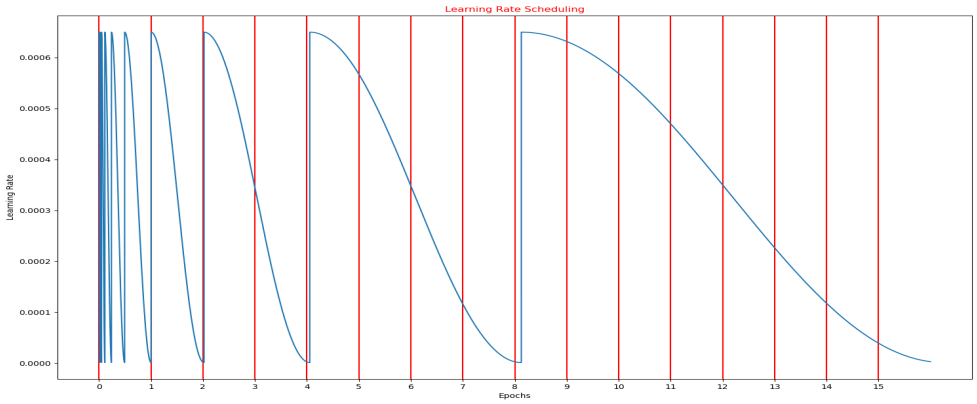
In the Stage-2, for a better generalization process there was added Stochastic Weighted Averaging (SWA).

Usually this process can be applied after the initial training stage, by using the last checkpoint. When the model converges we can restart training by setting the learning rate to a higher value and averaging the following checkpoints.

For this project we used SWA differently, instead of using the last checkpoint and retraining after it, we can use the checkpoints close to convergence during the initial training, this can be easily achieved by using Cosine Annealing Warm Restarts and choosing the the checkpoints when the learning rate is lower, right before restart.

As you can see in the image below, based on the learning rate scheduling procedure we can choose checkpoints at certain high learning rate values and average them together to obtain a better generalization for our models.

In our case, we used as checkpoints to average the weights of the model at the current epochs: $[2, 3, 4, 6, 7, 8, 12, 13, 14, 15, 16]$



Also, during Stage-2, based on the Grid Search from Stage-1, we can restrict the hyper-parameter space to find more specialized models and be able to focus to a specific region of the loss function.

Experimented Hyper-parameters during Grid Search (Stage-1):

- Optimizers: Adam, AdamW
- Learning Rate: 8e-6, 1e-5, 2e-5, 4e-5, 6e-5, 8e-5, 1e-4, 2e-4, 4e-4
- Learning Rate Scheduling:
    1. CosineAnnealingWarmRestarts (Optimal Parameters Chosen)
    2. OneCycleLR

Stage-2 Conclusions: Fixing the learning rate interval between 6e-5 to 65e-5, using Cosine Annealing Warm Restarts with optimal parameters according to batch-size, (for training stability) and replacing AdamW with Adam.

7 Pseudo-Labeling and Embeddings Extraction (Stage-3)

In the Stage-3 of this project we will develop the final models with the hyper-parameters selected from the experiments in the previous stages.

In order that our models to be trained on more data we switched from a 5 folds cross-validation scheme to a 10 folds cross-validation scheme, based on this we gained at training 10% more data for training.

As a final trick, based on the previous experiments, to be able to train the final models on even more data, we will use pseudo-labeling. We will take those samples from the test set that we are confident that our models can label them correctly.

Using pseudo-labeling can lead to noisy labels, for that reason we need robust rules for selecting and training on samples from the test set. In this particular case, we took all models with accuracy over 75% (35 models) and taking the samples for each of all selected models agreed on the label.

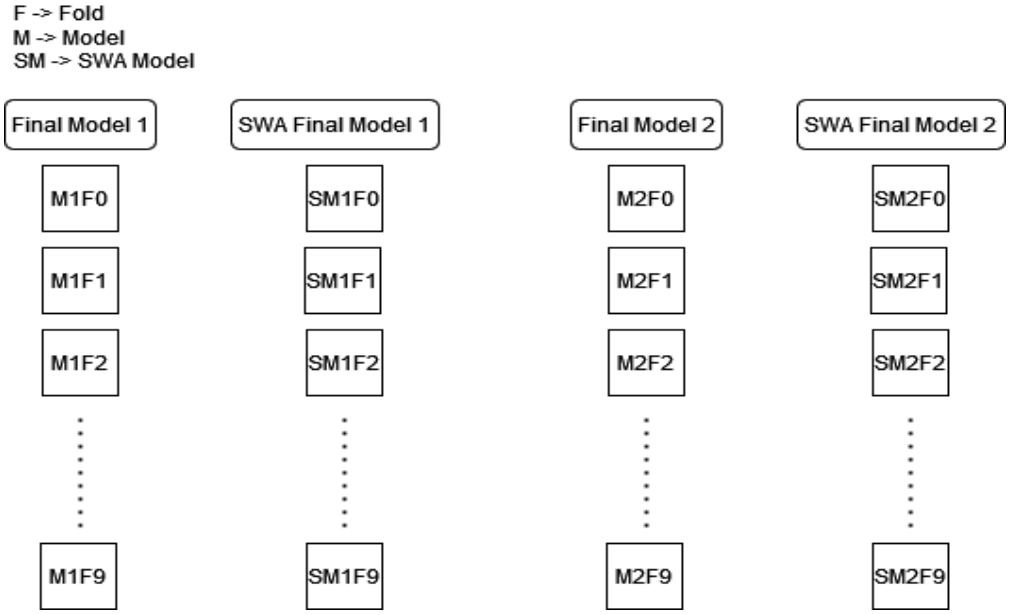
The number of those samples specified before is approximately 2300 samples from the test set.

To be sure that the idea of pseudo-labeling does not generate a data leakage issue, we will go further with two separate approaches one with pseudo-labeling and another one without.

Based on previous ideas, we have the following configurations for our models:

1. Final-Model-1
    - 10 Fold Cross-Validation
    - Backbone Architecture: Swin Large Transformer
    - Image Size: 384
    - Loss Function: Cross Entropy Loss
    - Optimizer: Adam
    - Learning Rate: 6e-5
    - Learning Rate Scheduler: Cosine Annealing Warm Restarts
    - Augmentation: False
    - Stochastic Weighting Averaging: True
    - Pseudo-Labels: False
2. Final-Model-2
    - 10 Fold Cross-Validation
    - Backbone Architecture: Swin Large Transformer
    - Image Size: 384
    - Loss Function: Cross Entropy Loss
    - Optimizer: Adam
    - Learning Rate: 65e-5
    - Learning Rate Scheduler: Cosine Annealing Warm Restarts
    - Augmentation: False
    - Stochastic Weighting Averaging: True
    - Pseudo-Labels: True

Notation



7.1 Final Models Results

Accuracy	Final Model 1	SWA Final Model 1	Final Model 2	SWA Final Model 2
Fold 0	0.766	0.767	0.767	0.774
Fold 1	0.768	0.778	0.773	0.774
Fold 2	0.770	0.772	0.774	0.778
Fold 3	0.761	0.771	0.771	0.763
Fold 4	0.753	0.754	0.759	0.772
Fold 5	0.770	0.775	0.779	0.783
Fold 6	0.768	0.772	0.766	0.778
Fold 7	0.767	0.755	0.767	0.759
Fold 8	0.768	0.773	0.770	0.775
Fold 9	0.761	0.768	0.761	0.761
Mean	0.765	0.769	0.769	0.772

7.2 The Last Submissions and Overfitting the Public Leaderboard

The last 3 days, represented the final experiments on the public leaderboard, from which the final submissions should have been selected.

In day one, to be sure that we avoid any leakage by pseudo-labeling we will use only the first model with the SWA variation of it.

The first submission will be represented by a voting system with all 20 folds (10 from standard model, 10 from SWA model).

Results: Public Leaderboard 0.81090, Private Leaderboard: 0.78230 ✓

The second submission will be represented by a voting system with only the best candidates for each fold, 10 folds in total.

Results: Public Leaderboard 0.80145, Private Leaderboard: 0.78133

In day two, we will add to the ensemble the model trained with pseudo-labeling and the SWA variation of it.

The first submission will be represented by a voting system with all 40 folds (10 from standard Final Model 1, 10 from SWA Final Model 1, 10 from standard Final Model 2, 10 from SWA Final Model 2).

Results: Public Leaderboard 0.80509, Private Leaderboard: 0.78739

The second submission will be represented by a voting system with only the best candidates for each fold, 10 folds in total.

Results: Public Leaderboard 0.80727, Private Leaderboard: 0.78836

7.3 As a final touch

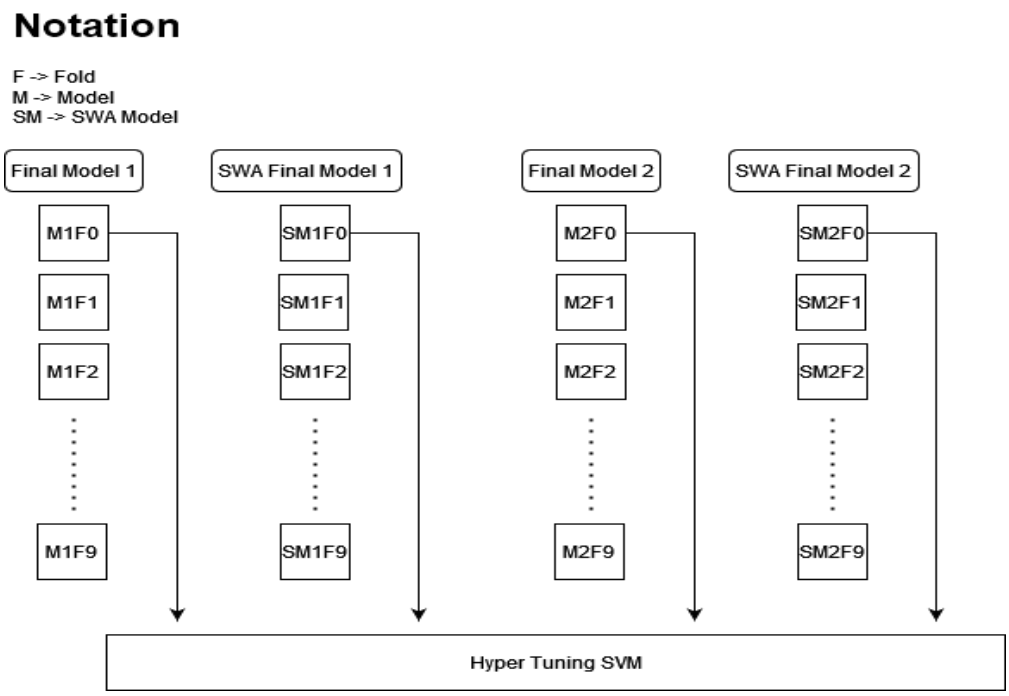
In order to ensure enough variance in our ensembles we will use only the backbone architectures for generating embeddings of those images, which will be classified later by an optimized SVM.

7.3.1 Multiple Concatenated BackBones with SVM Head

We can get a better intuition around the idea from the diagram below.

- We generate embeddings with each backbone architecture
- We recreate the same cross-validation split (to avoid validation leakage)
- We concatenate the embeddings
- We train an SVM over the whole feature space.

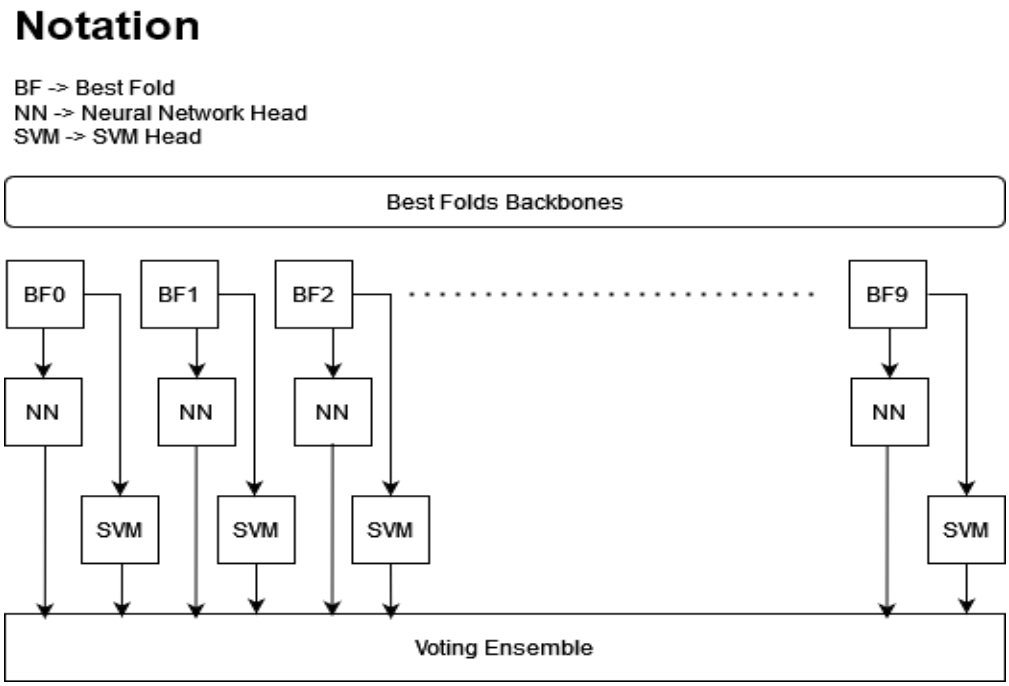
This idea would be impracticable at the training stage because it would require significant GPU Memory. Based on this we can generate a lower level ensemble, generating new 10 Folds trained on the learned patterns of each model and combining them.



This represented the first submission in the third and last day.

Results: Public Leaderboard 0.81090, Private Leaderboard: 0.78715 ✓

And the final submission was training only on the embeddings for the best fold, generating 10 another folds with SVM head and adding to the ensemble with the best folds with neural network head. Check diagram below.



Results: Public Leaderboard 0.80363, Private Leaderboard: 0.78884

Unfortunately, my best submission was not selected for the final evaluation.

The last of the three solutions selected for the final evaluation was an old ensemble with the following results

Results: Public Leaderboard 0.81090, Private Leaderboard: 0.78763 ✓