# Solution Guide

## Approach Summary

Create an ensemble from a library of diverse models with different architectures and augmentations. All models are initially pre-trained on imagenet and fine-tuned on the dataset. The models and augmentations are chosen automatically using hyperparameter optimization.

## Data Preprocessing

1. Rotate each image to its original view by reading the metadata of the image.
2. Resize each image to 512X512
3. Change any Grey image to RGB

## Model Architectures

The following architectures are included in the search space of hyperparameter optimization to choose from:

* ResNet [1]
* ResNext [2]
* WideResNet [3]
* DenseNet [4]

## Data Augmentations

The following augmentations are included in the search space of hyperparameter optimization to choose from:

* Rotation
* Random cropping and resizing
* Horizontal flipping
* Vertical flipping
* Brightness augmentation
* Hue augmentation
* Contrast augmentation
* Mixup augmentation

## Common Configurations

The following configurations is applied on all trails in hyperparameter optimization process :

* Stochastic Gradient Descent (SGD) optimizer
* Snapshot ensemble [5]
* 5-Fold training

## Ensemble Selection [6]

After generating the library of models, Ensemble Selection algorithm is applied to find the best combination of models with the lowest validation error.

## Post Processing

Samples with high prediction confidence (>0.7 softmax probability) for the healthy wheat class is considered 1.0

## Project Files

* train\_data.zip: original contest’s training data
* test\_data.zip: original contest’s test data
* prepare\_dataset.py: reads training and test data, removes duplicates from training data and saves them in numpy matrices. It outputs the following:
  + ***unique\_train\_imgs\_rot\_fixed.npy***: numpy matrix with shape (732, 512, 512, 3) of unique training images after resizing to 512X512
  + ***unique\_train\_gts\_rot\_fixed,npy***: numpy matrix with shape (732,1) of training ground-truth
  + ***test\_imgs\_rot\_fixed.npy***: numpy matrix with shape (610, 512, 512, 3) of test images after resizing to 512X512
  + ***test\_gts.npy***: numpy matrix with shape (610,1) of dummy test ground-truth (used just to unify the input to the dataset class but it is all zeros)
  + ***ids.npy***: numpy matrix with shape (610,1) of names of test images
* config.py: has the configurations of best single models manually created.
* dataset.py: has the dataset class for training and test data
* utils.py: utility functions for training, testing and reading dataset images
* generate\_library\_of\_models.py: generates a library of models with different architectures and augmentations through hyperparameter optimization search. It creates “***trails***” folder in which validation and test predictions of all trail models are stored.
* main.py: script for training a single model manually configured for k-folds style. It saves the validation and test predictions of the trained model in the “***trails***” folder. It has the following argument:
  + ***--config\_id***: configuration id (1, 2, 3, or 4). Each configuration chooses a specific architecture along with its training parameters.
* ensemble\_selection.py: applies Ensemble Selection algorithm on the generated library of models to find the best ensemble with the lowest validation error and use it to create the final submission. It outputs ***final\_sub.csv*** which has the test predictions.
* requirements.txt: has the needed python3 APIs used in this project.

## My Machine Specifications

* + AMD 3700X
  + 32GB Ram
  + 64GB Swap
  + Nvidia RTX 2080Ti GPU
  + Ubuntu 18.04
  + Python 3.6.9

## Used APIs

Check requirements.txt file.

## Running Steps

1. ***unzip train\_data.zip***
2. ***unzip test\_data.zip***
3. ***python3 prepare\_dataset.py***
4. ***python3 generate\_library\_of\_models.py***
5. ***python3 main.py --config\_id 1***
6. ***python3 main.py --config\_id 2***
7. ***python3 main.py --config\_id 3***
8. ***python3 main.py --config\_id 4***
9. ***python3 ensemble\_selection.py***

## References

[1] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[2] Xie, Saining, et al. "Aggregated residual transformations for deep neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[3] Zagoruyko, Sergey, and Nikos Komodakis. "Wide residual networks." arXiv preprint arXiv:1605.07146 (2016).

[4] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[5] Huang, Gao, et al. "Snapshot ensembles: Train 1, get m for free." arXiv preprint arXiv:1704.00109 (2017).

[6] Caruana, Rich, et al. "Ensemble selection from libraries of models." Proceedings of the twenty-first international conference on Machine learning. 2004.