# Solution Guide

## Approach summary

Crop a patch around each field and pass it to a deep neural network model for classification. The surrounding visual information around each field’s pixels is very useful to a deep model. Since the dataset is small and has a lot of classes, the model is designed to be parameter-efficient and carefully trained with extensive augmentation to avoid overfitting.

## Data Pipeline

### Data Generation

This step aims at creating a visual context around field pixels. The steps are:

1. Find the center pixel of each field.
2. Crop a 32X32 patch around the center for each band for each date so each field is represented by a matrix of size (T, C, H, W) where:
   * T: number of time steps = 13
   * C: number of spectral bands = 13
   * H: height = 32
   * W: width = 32
3. Crop a 32X32 field mask around the same center where field pixels are ones and others are zeros. The size of each field mask matrix is (1, 1, H, W).

### Feature Selection

Remove band B11. The matrix size for each field will be (13, 12, 32, 32).

### Feature Engineering

This step aims at adding relevant features to the task since it will be difficult to learn them with the limited available training data. Calculate the following vegetation indices for each date in a field:

* [1]
* [2]

The matrix size for each field will be (13, 15, 32, 32).

### Data Preprocessing

This step aims at normalizing the input data to ease the convergence of neural network training. The steps are:

1. Apply sqrt to decrease skewness
2. Apply standard normalization to bound the input range

### Data Augmentation

The following augmentations are applied randomly in the training to reduce overfitting:

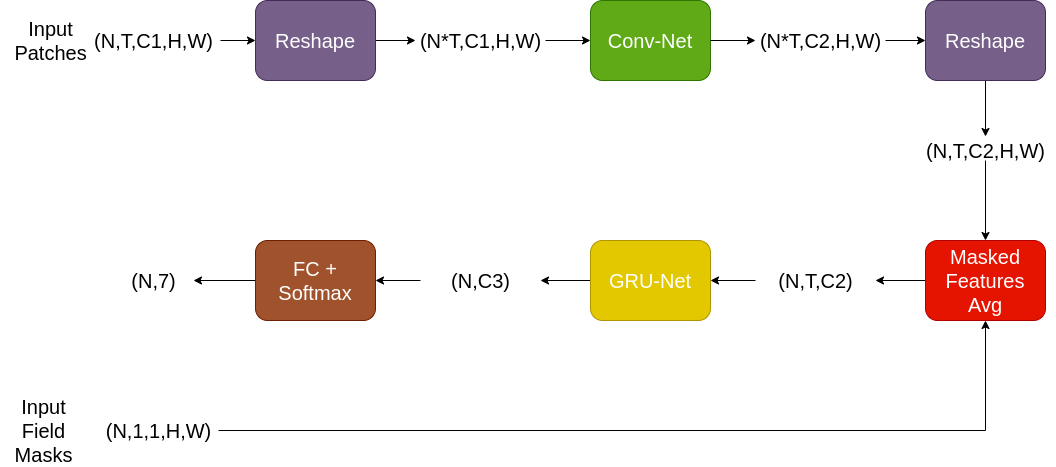
* Spatial augmentations: rotation, vertical flippling, random cropping
* Mixup: select a random crop from any of the 4 big tiles and add a fraction of it to the input patch.
* Time augmentation: drop randomly one time step.

## Model Pipeline

### Model Architecture

It consists of three parts:

* Conv-Net: 3 conv layers followed by group-norm and leaky relu. The same network is applied on all time steps.
* Masked Features Average: averaging of pixels only belongs to each field.
* GRU-Net [3]: 3 bi-directional gru layers applied on the averaged features of time steps.
* Fully connected + softmax: final classification layer.



### Training Parameters

* + SGD optimizer
    - Momentum: 0.9
    - Learning rate: 0.008
  + Snapshot ensemble [4]:
    - 6 cycles
    - 10 epochs per cycle
  + Batch size: 16

### Ensemble

* + Average ensemble of 10 models of the same architecture each trained on a different subset (85%) of the training data
  + Average of the same pipeline run using the described architecture and a smaller version with 2 conv layers and 1 gru layers and trained with less augmentation.

## Project files

* + data.zip: compressed file of original contest’s training and test data.
  + main.py: starts the training and generates testing results for the small and large architecture. The output of this file is a submission file named either ***best\_config\_sub.csv*** or ***simple\_config\_sub.csv*** depending on the selected architecture. It has the following arguments:
    - ***--data\_path***: path to data folder. Default is “data”
    - ***--crops\_path***: path to generated crops. Default is “.”
    - ***--sample\_sub\_path***: path to sample submission file. Default is “SampleSubmission.csv”
    - ***--save\_path***: save path for output submission file. Default is “.”
    - ***--best\_submission***: True for enabling training the model with the best configuration and False for the simpler one. Default is False.
  + prepare\_data.py: reads the dataset tiles and generates the needed crops around each field. It outputs the following:
    - ***imgs\_13\_ch\_rad\_16\_medianxy.npy***: numpy matrix with shape (4688,13,13,32,32) which has the crops for training and testing field.
    - ***field\_masks\_medianxy.npy***: numpy matrix with shape (4688,13,13,32,32) which has the field mask of each crop for training and testing field.
    - ***areas.npy***: numpy matrix with shape (4688,1) which has the area of each training and testing field.
    - ***gts.npy***: numpy matrix with shape (4688,1) which has the ground-truth class of each training and testing field.
    - ***fields\_arr.npy***: numpy matrix with shape (4688,1) which has the id of each training and testing fields.

The file has the following arguments:

* + - ***--data\_path***: path to data folder. Default is “data”
    - ***--save\_path***: save path for output submission file. Default is “.”
  + model.py: has the classes for the two architecture used in the ensemble
  + dataset.py: has the class for dataset. Augmentation is included in this class.
  + utils.py: utility functions for training, testing and tiff image reading
  + Combine\_subs.py: scripts for combining the results of small and large models and outputs final submission file ***final\_sub.csv***. It has the following arguments:
    - ***--subs\_path***: path to the folder that has both ***best\_config\_sub.csv*** and ***simple\_config\_sub.csv*** files.
    - ***--save\_path***: save path for output submission file. Default is “.”
  + 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:

The following steps assume data and save path are in the same folder as the code files. You can change these paths as you wish.

1. ***unzip data.zip***
2. ***python3 prepare\_data.py***
3. ***python3 main.py***
4. ***python3 main.py --best\_submission True***
5. ***python3 combine\_subs.py***

## References

[1] A. Karnieli, N. Agam, R. T. Pinker et al., “Use of NDVI and land surface temperature for drought assessment: merits and limitations,” Journal of Climate, vol. 23, no. 3, pp. 618–633, 2010.

[2] S. K. McFeeters, “The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features,” International Journal of Remote Sensing, vol. 17, no. 7, pp. 1425–1432, 1996.

[3] Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).

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