**Network Architecture**

The network consists of a contracting path (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional neural network. It consists of a repeated application of two 3x3 padded convolutions (specifically a pad of one), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride of 2 for down-sampling. At each down-sampling step, we double the number of feature channels. Every step in the expansive path consists of an up-sampling of the feature map followed by a 2x2 convolution (up-convolution) that halves the number of feature channels, a concatenation with the correspondingly same-sized feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU function. **Before the feature map gets to the each ReLU function for the channel output, batch normalization is used to improve the network’s learning rate. A padding of one prevents the input feature from being cropped which makes the network work good irrespective of the image size used for training**. At the final layer a 1x1 convolution is used to map each 64- component feature vector to the desired number of classes. In total the network has 23 convolutional layers. In general, the whole network has a UNET architecture.

**Dataset**

Out of the total 129 images that were provided for training. 23 were images (id-0 to id-22) did not have any shape files for extraction using ArcGIS Pro. Coincidentally, these 23 images are the images for the second challenge. 8 images are not extracted at all as I could not find their where-abouts. The images became a bit difficult to track as time elapsed. Out of the remaining 98 images, 24 are very noisy and there are intentionally not extracted. Noisy images fall under these categories:

1. Linescans that had a lot of black areas in the image and were not clear enough.
2. Linescans that had their mask only in black of the image after polygon extraction.
3. Linescans that did not match their masks after side-by-side comparison.

These images are avoided to prevent the model from always learning that black areas are areas of fire.

Of the remaining 74 images, 39 images are used for data augmentation for training. 10 are used for the testing set. 25 of the remaining datasets are used validation of the model.

**Feature Engineering, Feature Selection & Training**

As the training image sizes varies, the model is trained with 512x512 sized images. The UNET model has been trained at a learning rate of 0.004 with 2 workers to simultaneously feed data to the RAM. As a result of noise in the data, only 49 carefully selected images with no or less are selected for training as **explained in the dataset section**. 39 are used for data augmentation. In each batch, five images are loaded and after three transformation events (rotation – 100% chance, horizontal flipping – 50% chance, vertical flipping – 10% chance), there is a 0.0275 chance of the same transformation recurring in each batch. As a result, enough data were engineered for training to avoid **overfitting as supported by the graph below.** Pin memory enabled speeds up the transfer of data to GPU for faster training. The model runs for 200 epochs with five new augmented images during each epoch.

**Model Validation**

As mentioned in the dataset section above, 74 images are used in the model from training to validation. 39 images are used for data augmentation for training. These 39 images are carefully selected from all fire events. These images have less or no noise. The model selected five images randomly for data augmentation, and therefore the model had enough variant images for training. Data augmentation happens during the batch loading process, and not separately (not before model begins). 10 images are used for the testing set. This formed 15% of the whole set of images used for training. 25 of the remaining datasets are used validation of the model after training (some of these 25 images are very noisy and are there on purpose to see how the model performs on noisy images).

**To minimize the overhead and make maximum use of the GPU memory, we favor large input tiles over a large batch size and hence reduce the batch to a single image.**