**Approach**

As explained in the [video](https://drive.google.com/file/d/191v0Qa-jBUbY_wmYW2IV-GiUNPv3jmmQ/view?usp=sharing), the code has been hosted on github. To save you from the installation of a couple of libraries, run the code on google colab which already has all needed packages installed.

The first thing to do after cloning the code is to separate your animated-gif you get from the jupyter notebook online into the individual frames. After getting the frames, convert them to a single channel and get the image segmentations using the model from challenge 1. Replace the colored separated frames in the test and train folders with the black and white masks of the same frames.

Run the training cell. This trains the model using the prior images before the next frame prediction. After training, use the last saved model check point for the next-frame prediction.

To decide on which specific frame will be the fire prediction on a specific date, I used the average days, hours or minutes between the timestamps. For example, for the rosedale fire event, I calculated the average minutes between the separate fire events to be 8minutes. As such, I selected the 5th and 33rd frames for the rosedale fire events. The next frame prediction comes with the same resolution the challenge 2 jupyter notebook gave online. As such, using coreldraw, I used the size ratios from the last fire event before the first next-frame fire prediction to rescale the prediction to the actual size. Check the coreldraw files for better understanding. Also, I used coreldraw to redraw the predictions as they were not as solid in resolution as expected.

After getting the right size of the prediction, the image is loaded into ArcGIS Pro. The 1000 test coordinates are then converted to the right coordinates system and plotted on the predicted images. Points that lie are on the white sections are then selected as areas with fire. Since the calculation of average days, hours or minutes does not take into consideration of many natural factors, I selected points a few meters away or closer to the white sections of the prediction. This improved the accuracy of the first and third nunnet events, and for the rosedale event. For the first nunnet event. The prediction was correct; moving points further or closer rather decreased the accuracy.

In summary, the next frame predictions from the model miss some points (50-100) around the predicted area so please pay attention to that.

**Network Architecture**

The model uses the Pix2Pixmodel as its baseline model. The Pix2Pix model uses a cGAN (Conditional Generative Adversarial Network) model for image-to-image translation. It comprises of a generator model and a discriminator model. The model is trained on a set of paired datasets, i.e. a semantic label map and the corresponding real image. U-Net serves as its generator which outputs realistic-looking images from the semantic label maps. It uses a patched based fully convolutional network as its discriminator which detect whether the output image is real or not. The generator is trained on a traditional GAN loss function. Loss is measured between the generated imaged and its expected output.

**Modifications to the Pix2Pix Model**

To improve the generator and discriminator architecture, a Course to fine generator and a multi scale general discriminator is used.

**Course to fine generator**

* This is made up of two components, the global generator network and local enhancer network.
* The global generator architecture – This has three components: a convolutional front-end, a set of residual blocks, and a transposed convolutional backend.
* The local enhancer network architecture – This also consists of 3 components: a convolutional front-end, a set of residual blocks, and a transposed convolutional backend.

**Multi-scale discriminator (still based on the patched based fully convolutional network)**

* Three discriminators (D1, D2 & D3) with identical network structures but different image scales are used.
* Both real and generated images are down sampled by a factor of 2 and 4. The discriminators are then trained to differentiated between real and synthetic images at 3 different scales. (i.e., D1 at the original size of the images, D2 at the images scaled down by a factor of 2 & D3 at images scaled sown by a factor of 4).

**Improved adversarial loss**

* Feature matching loss based on the discriminator is used.

**Feature Engineering, Feature Selection & Training**

To avoid any colorful noise in the predicted next-frame images as [these](https://drive.google.com/drive/folders/1Dekt-ncj6WSe2lQTaQggBfTsGFCF6_kV?usp=sharing), I decided to use the mask of the images obtained from the challenge 1 prediction rather for training. Unfortunately, sometimes, the challenge 1 model sometimes predicts with noise around it. As such, I redrew the images with coreldraw in such cases. I realized the generator picks up every detail on each frame for the next-frame prediction, so I used on frames that had consistent fire advancement. Such was the case for the first nunnet fire event. I used only 5 frames out of the 7 frames to get a better prediction. Specifically, I ignored the [first](https://drive.google.com/file/d/16kFH5-mabjNo4c_dOsXGnrYWDfeD4pXt/view?usp=sharing) and [fifth](https://drive.google.com/file/d/1DActyVkHqTIIUklUWVCOIH3SFElxD236/view?usp=sharing)frames. The discrepancies between the frames were too high, so I intentionally avoided them.

Training the model with the immediate frames before the next frame prediction for 200 epochs at a learning rate of 0.002, the improved adversarial loss encourages the generator model to create plausible translations of the source image.

**Model Validation**

The model is always trained with each fire event’s frames before the next frame prediction can be obtained. With only these few frames for training and testing, only the last frame in set of frames is used for testing; the rest are used for training. After training, setting the video frame rate and number of frames produces enough subsequent frames as needed.

**Performing features**

Using the colored frame for prediction really creates noisy next-frame predictions. As such, use the masks from the first model rather for the next frame prediction. There aren’t any importance plots, but this [folder](https://drive.google.com/drive/folders/1Dekt-ncj6WSe2lQTaQggBfTsGFCF6_kV?usp=sharing) draws more attention on using on the mask as a great performing feature.

**Evolution of our approach**

I realized using all the frames for training at once and making predictions does not give good results. So, I trained the model on three occasions with each fire event’s frames, and the results were better.

Also, I realized using the colored frames for prediction gives bad predictions as the frames itself contain noise (the black areas as explained in the challenge 1 document). Using the mask which contains just the black and white colors rather gives better results.

Finally, after getting predictions, I realized the model missed 50-100 points away or closer to the predicted white areas. I therefore selected a few points away or close to the white areas. This increased the accuracy in some of the fire-events.

**Unique idea about approach**

The model is always trained on the prior frames before the next-frame prediction. This encourages the generator model to create plausible translations of the source image directly. As such, the model picks up every detail in the prior frames for the prediction. The next-frame predictions are therefore always close the required targets.

**Limitations**

One great limitation but a good process is always training the models on the prior images. This takes some time. However, it is a good process because the generator always produces results closer to the target.

Also, the resolution of the predictions is low so you need to redraw them for a good and solid output. Again, you need to resize the image to suit the original image’s size.

**Optional Requirements**

One element I would like to highlight is the resizing of the predicted images to suit the original image’s width and height. Use the normal ratio matching of the height to height of original’s image height to next-frame prediction’s height. Do the same for the width. Just make sure the enlarge touches two of the enlargement’s edges as illustrated in the coreldraw file.

One factor that caused a breakthrough was using the mask for training instead of the colorful images.

Training the model with each fire event’s frames takes some time as stated above. That is a challenge because for the second and third nunnet fire event with 13 frames, it took 104mins in total to do training before prediction could be made.

For prediction time, after model has been trained, 60 next frame predictions take less than 30s.