Using PyTorch to Predict Wildfires

Bob Chesebrough, Rahul Unnikrishnan Nair

intel®

Agenda

- Introduction to the topic
- Lab setup
- Conceptual process: getting real data, labeling with view to predicting danger zones 2 years in advance
- Overview of finetuning with PyTorch*, ResNet 18
- Introduction to Intel® Extension for PyTorch*
- Using synthetic data – Stable Diffusion*
- Accelerating Stable Diffusion pipeline with Intel® Extension for PyTorch*

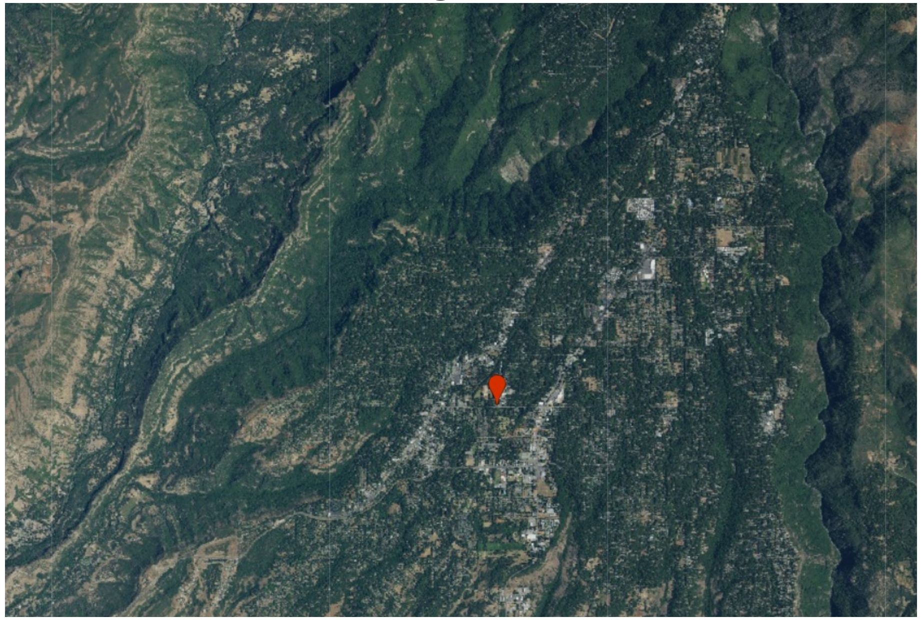
Forest fires

- Loss of human lives
- Damage to ecosystems and wildlife
- Early identification of high fire likelihood
  - Allows time for remediation
  - Allows precious resources to be used wisely
- AccuWeather Founder and CEO Dr. Joel N. Myers stated that the "total damage and cumulative economic loss for the 2021 wildfire season was expected to be between $70 billion and $90 billion in the U.S. with $45 billion to $55 billion of those damages to California alone".

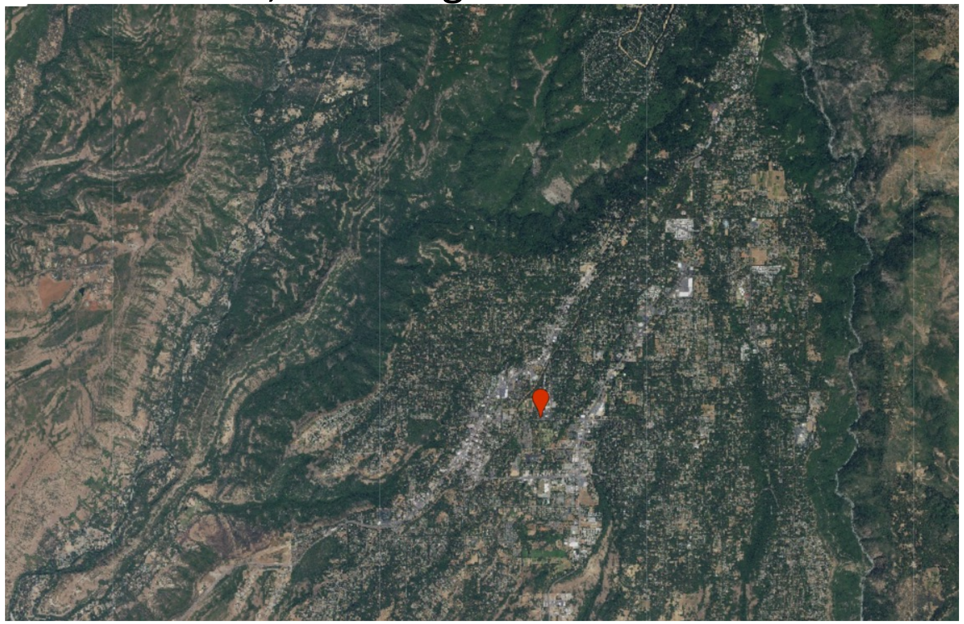
Risk Assessment from Aerial Photos analysis

- Aerial photos (or satellite images)
  - Geo-spatial information and elevation gain/loss can inform fire likelihood
  - Color offers hints to health and density of foliage

Paradise, CA: Image BEFORE 2018 Fire



Paradise, CA: Image AFTER 2018 Fire



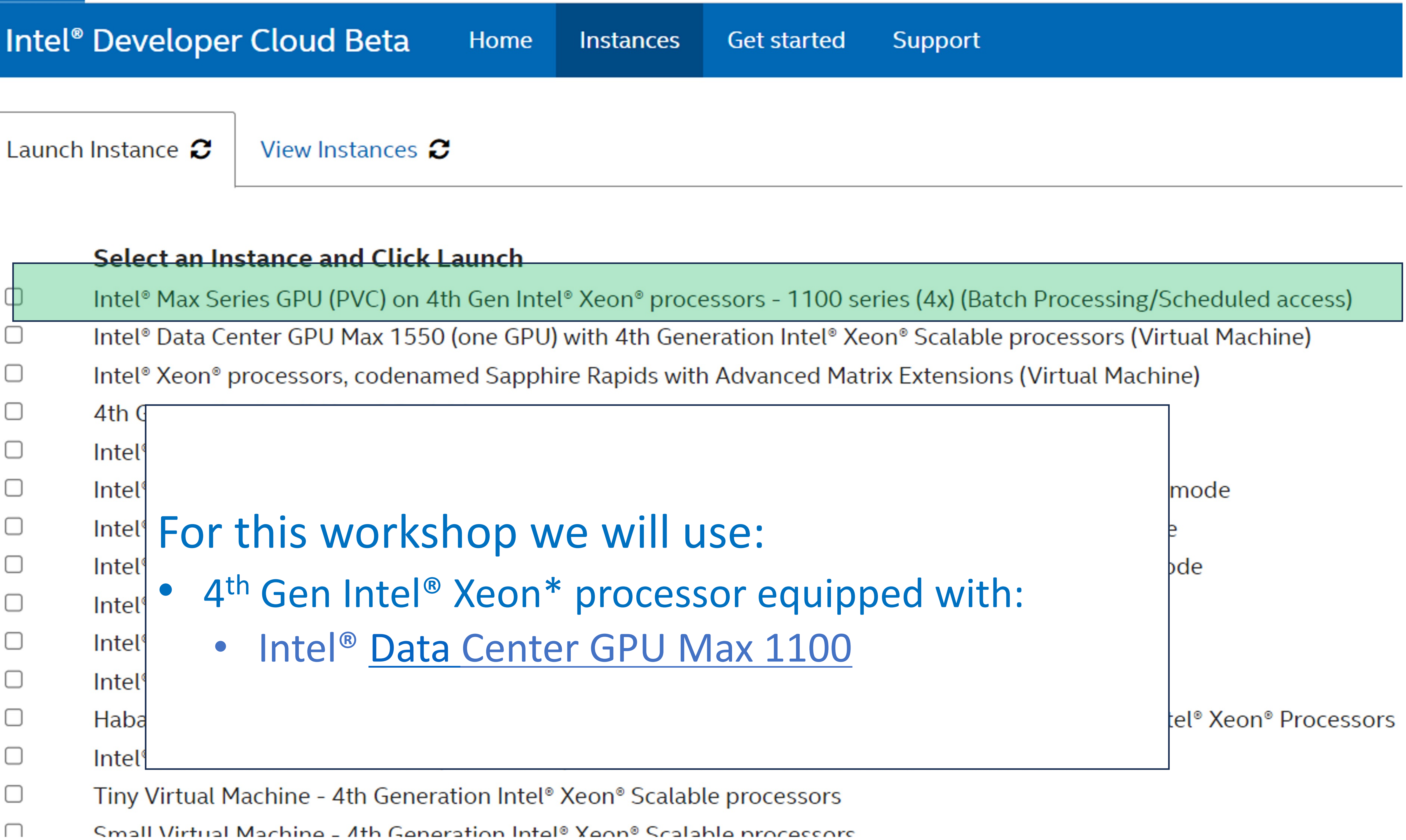
(dataset) Aerial Photography Field OfficeAPFO)2021). NAIP Digital Ortho Photo Image Geospatial_Data_Presentation_Form: remote-sensing image. West Virginia GIS Tech Center. https://data.nal.usda.gov/dataset/naip-digital-ortho-photo-image-geospatialdatapresentationform-remote-sensing-image. Accessed 2023-07-22.

Lab

- Follow me!

Intel® Developer Cloud Beta

- Instructions for access:
  - Intel® Developer Cloud Beta: https://github.com/bjodom/idc
  - Code and Data: https://github.com/IntelSoftware/ForestFirePrediction



For this workshop we will use:

- 4th Gen Intel® Xeon* processor equipped with:
  - Intel® Data Center GPU Max 1100

Predicting Forest Fires: Data

- Using aerial photos is interesting but prediction implies a time component
  - Possible Sources of data (USDA/NAIP/DQQQ dataset)
    - Direct Download: https://datagateway.nrcs.usda.gov/
    - Google Earth Engine (use to search by time and place and to render image)
    - ARGIS (use to search by time and place and to render image)
    - https://earthexplorer.usgs.gov/ (use to filter and download geotiff images)

Predicting Forest Fires: Location & Time

Google Earth Engine Viewer for USDA/NAIP/DQQQ

Javascript to generate arial photo from given time window for given map center

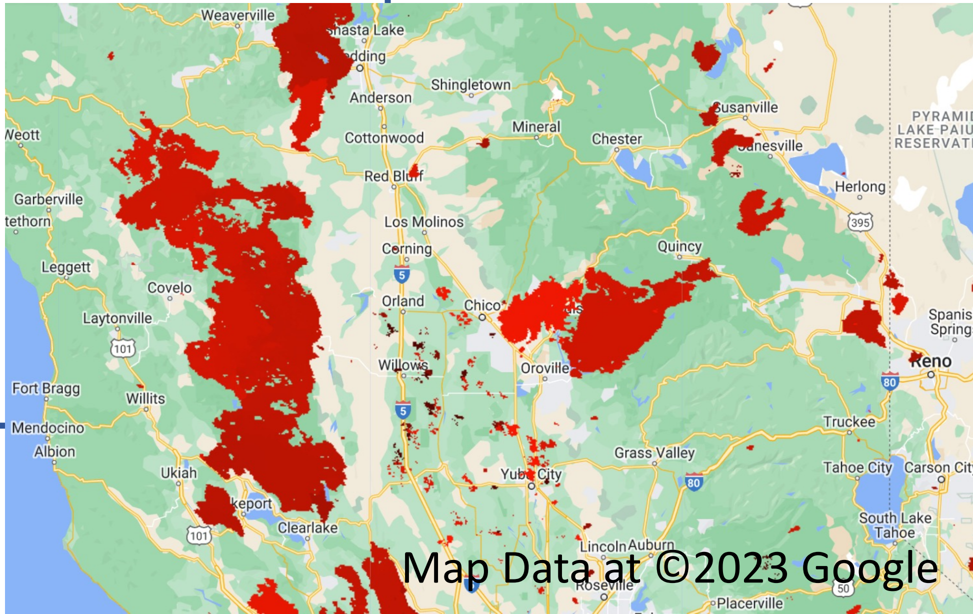
```
var dataset = ee.ImageCollection('USDA/NAIP/DOQQ')
                  .filter(ee.Filter.date('2018-01-01', '2020-01-01'));
var trueColor = dataset.select(['R', 'G', 'B']);
var trueColorVis = {
  min: 0.0,

  max: 255.0,
};
Map.setCenter(-121.614,39.76, 13);
Map.addLayer(trueColor, trueColorVis, 'True Color');
```

- Example modified from example shown here: https://developers.google.com/earth-engine/datasets/catalog/USDA_NAIP_DOQQ

Predicting Forest Fires: Labeling

- Determining historical forest fire locations
- Use NASA: MODIS/006/MCD64A1 dataset
- Sample USDA/NAIP aerial photos within the boundaries
- Red region: Fire between 2018 - 2020

```
var dataset = ee.ImageCollection('MODIS/006/MCD64A1')
                  .filter(ee.Filter.date('2018-01-01', '2020-12-31'));
var burnedArea = dataset.select('BurnDate');
var burnedAreaVis = {
  min: 30.0,
  max: 341.0,
  palette: ['4e0400', '951003', 'c61503', 'ff1901'],
};
Map.setCenter(-121.62520673097843,39.77606238136723, 8);
Map.addLayer(burnedArea, burnedAreaVis, 'Burned Area');
```



Modis Fire Data, California near Sacramento and Chico. Google Earth Engine with 'MODIS/006/MCD64A1'

Predicting Forest Fires: Labels

- Use NASA: MODIS/006/MCD64A1 dataset
- Download Images from locations indicated by the pins
- place into “Fire” or ‘No Fire” folders
- Burn Area indicated in red – 2018 to 2020
- **Images** to train are from **2016 to 2017**

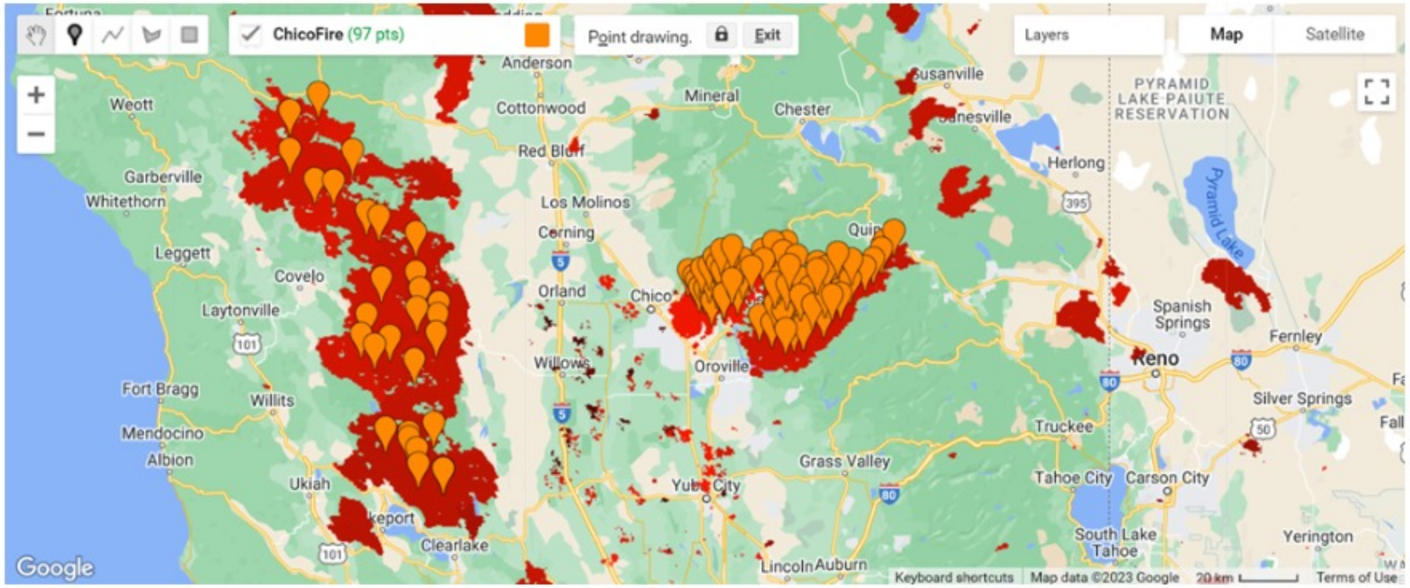


Figure 2. Sampled known fire locations used. Google Earth Engine with ‘MODIS/006/MCD64A1’ dataset

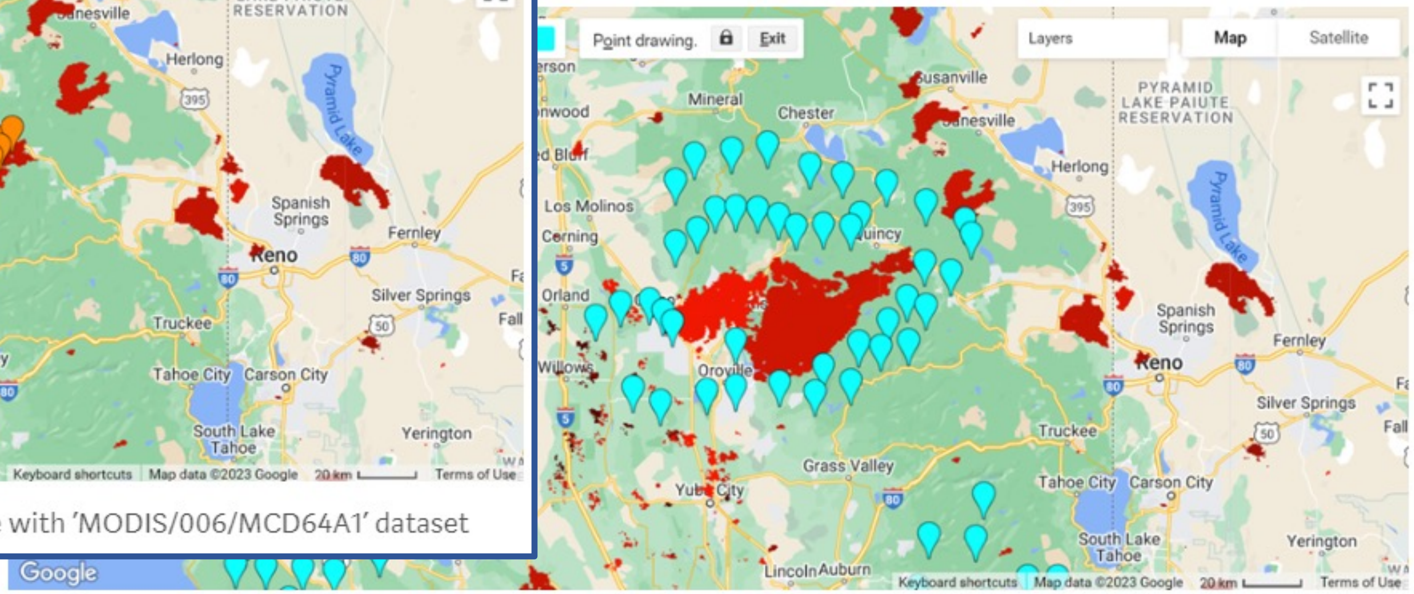


Figure 3. Sampled no fire locations used: Google Earth Engine with ‘MODIS/006/MCD64A1’ dataset

PyTorch*

- Used PyTorch* 1.13 and Torchvision* ResNet* model as base
  - Configurable but we used ResNet 18)
- Sample a couple hundred images
- Optimize with Intel® Extension for PyTorch*
  - Convert model
  - Convert data
- Use Finetuning
  - On CPU: Finetuning on ResNet 18 is fast – takes a few minutes to get a preliminary model
  - On XPU: Finetuning on ResNet 18 is fast – takes a few minutes to get a accurate model

Intel® Extension for PyTorch*

- Extends PyTorch* with up-to-date features optimizations for an extra performance boost on Intel hardware
- Optimizations take advantage of AVX-512 Vector Neural Network Instructions (AVX512 VNNI)
- Intel® Advanced Matrix Extensions (Intel® AMX) on Intel CPUs
- Intel X^e Matrix Extensions (XMX) AI engines on Intel discrete GPUs.
- Provides easy GPU acceleration for Intel discrete GPUs with PyTorch*.
- Installation via github

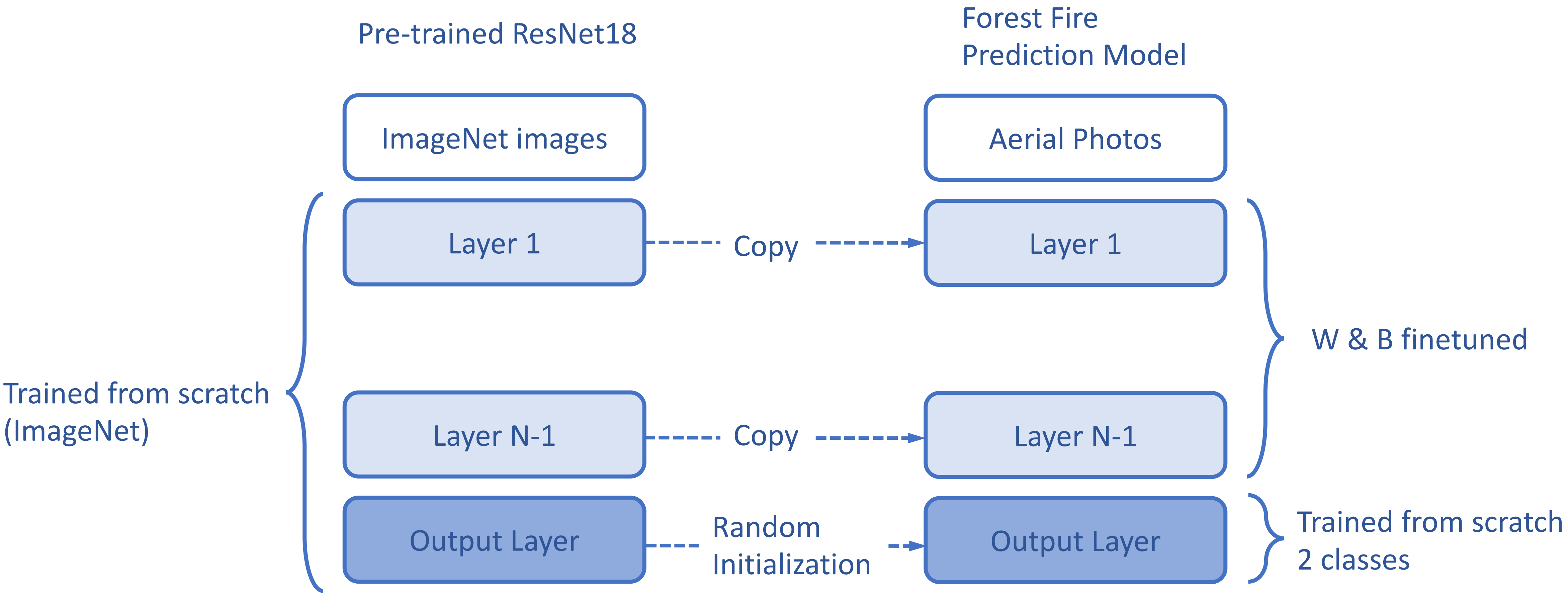
Intel® Extension for PyTorch*

Inference on CPU	Inference on XPU

<pre>import torch
import torchvision.models as models
 	
model = models.resnet50(pretrained=True)
model.eval()
data = torch.rand(1, 3, 224, 224)
 	
import intel_extension_for_pytorch as ipex
model = ipex.optimize(model)
 	
with torch.no_grad():
  model(data)</pre>
	<pre>import torch
	import torchvision.models as models
	
	model = models.resnet50(pretrained=True)
	model.eval()
	data = torch.rand(1, 3, 224, 224)
	
	import intel_extension_for_pytorch as ipex
	model = model.to('xpu')
	data = data.to('xpu')
	model = ipex.optimize(model)
	
	with torch.no_grad():
	  model(data)</pre>

Finetuning

- Target datasets are much smaller than source datasets
- Fine-tuning helps to improve models' generalization ability.
- Can be trained faster with fewer and less expensive compute



Code: High level

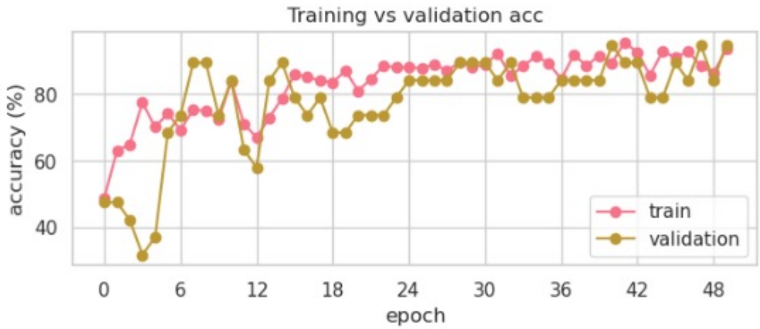
```python
# Import the Intel Extension for PyTorch library, which provides optimizations for Intel architectures
import intel_extension_for_pytorch as ipex
# Import the required modules from PyTorch and torchvision for deep learning tasks and dataset handling
import torch
import torch.nn as nn
import torchvision.models as models
from torch.utils.data import DataLoader
from torchvision import datasets, models, transforms
```

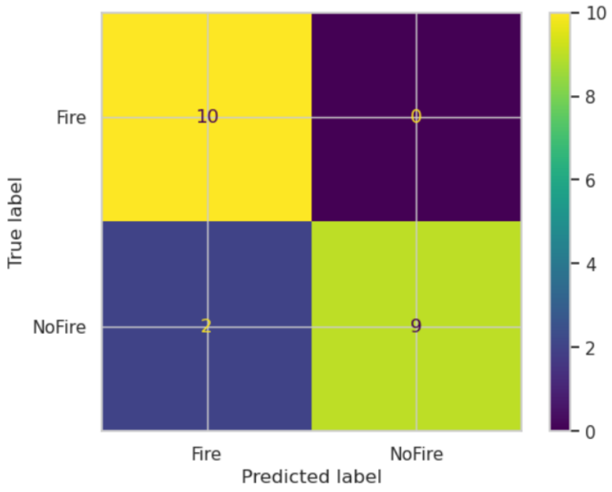
```python
def _to_ipx(self):
    """convert model memory format to channels_last to IPEX format."""
    self.model.train()
    self.model = self.model.to(memory_format=torch.channels_last)
    self.model, self.optimizer = ipex.optimize(
        self.model, optimizer=self.optimizer, dtype=torch.float32
    )
```

```python
time_per_epoch = []
if self.ipx:
    self._to_ipx()
print(f"fine tuning model for max epochs: {self.epochs} : lr = {self.lr}")
for epoch in range(self.epochs):
    print(f"Epoch: [{epoch+1}]")
    st_time = time.perf_counter()
    t_epoch_loss, t_epoch_acc = self.train()
```

torch version: 1.13
IPEX version: 1.13
Must match!

Results





- I only use on the order of 100 images per class
- so -> my holdout test set is small

```
loss:
 training set  : 0.1879
 validation set: 0.1953
accuracy:
 training set  : 0.9354
 validation set: 0.9474
time elapsed: 2408.4395961761475
```

	Train	Val	Test (unknown)
Fire	87	9	10
NoFire	90	10	11

Workshop Data: Stable Diffusion

- Synthesized data using Stable Diffusion
  - Algorithm to turn text into images
- We Optimized it using Intel® Extension for PyTorch* (IPEX)
- Try the lab to see the performance - wow ☺

Where did we get the data?

- Primarily from two US government sources:

1. NASA/MODIS Burn Area
2. USDA/NAIP/DOQQ Aerial photos

How to acquires these?

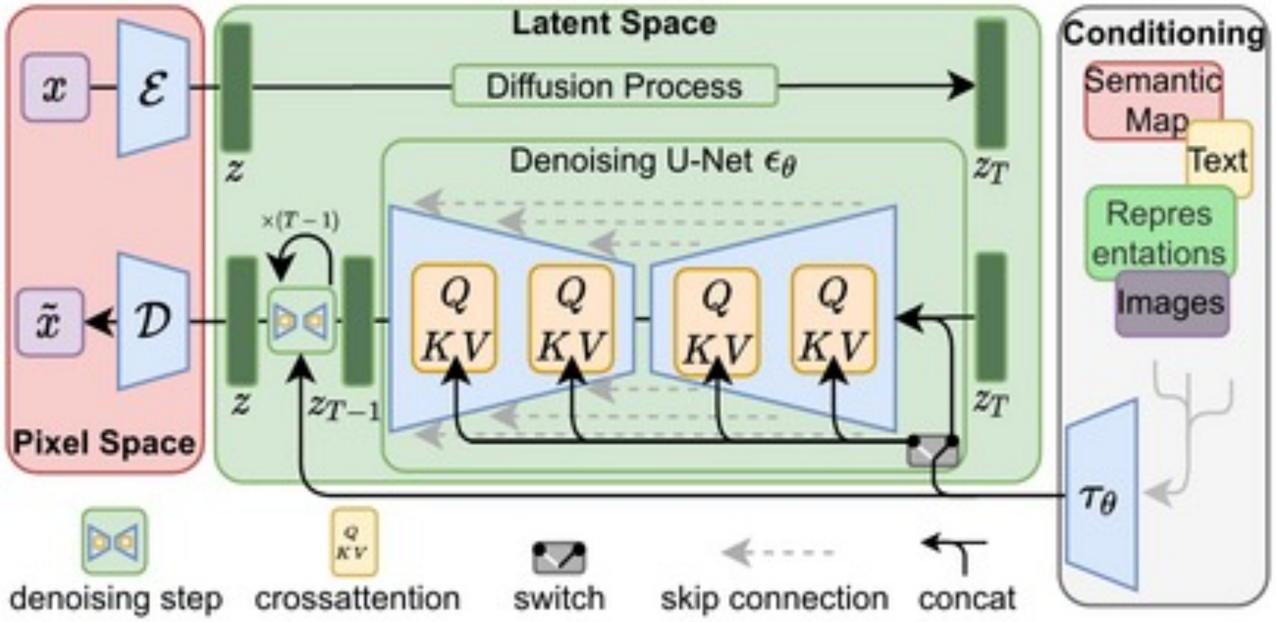
1) Google Earth Engine (best images – True Color setting): Not free*
2) USGS Earth Explorer- Free* for these images
3) ArcGIS: Not Free*

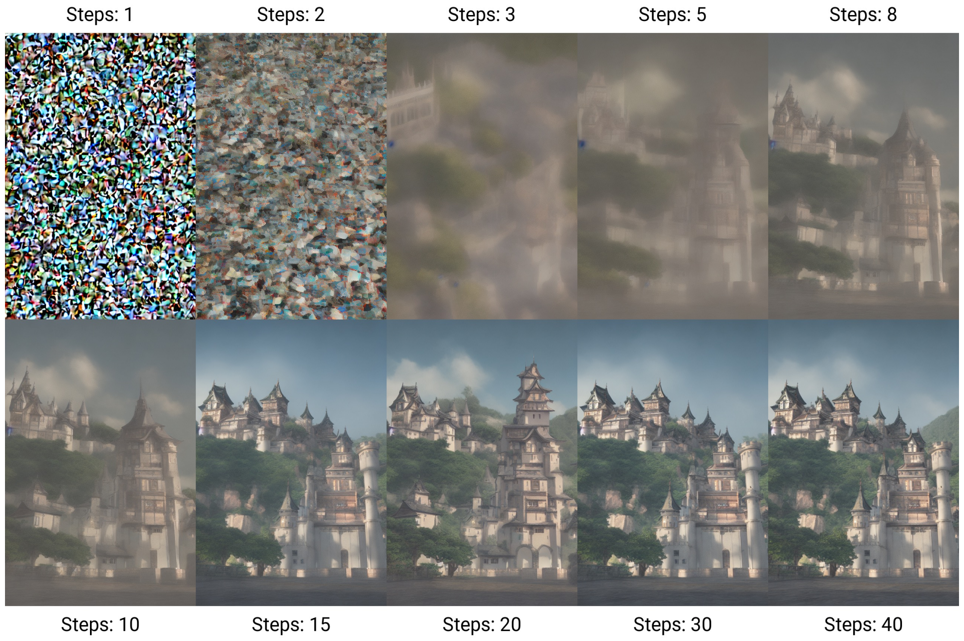
What data are you using in the workshop?

- Data from the USGS/Earth Explorer data (about 100 images total)
- Synthesized data generated with Stable Diffusion

- This allows us to demonstrate the principles used!
- At the expense of not being allowed to provide the best data for actual wild fire images from Google Earth Engine
- Here is a link to my blog on how to get real data and create more realistic forest fire prediction model
- “Predict Forest Fires using PyTorch”

What is Stable Diffusion





What is it? An image to image or text to image generator

What can it do?

Stable Diffusion on an XPU

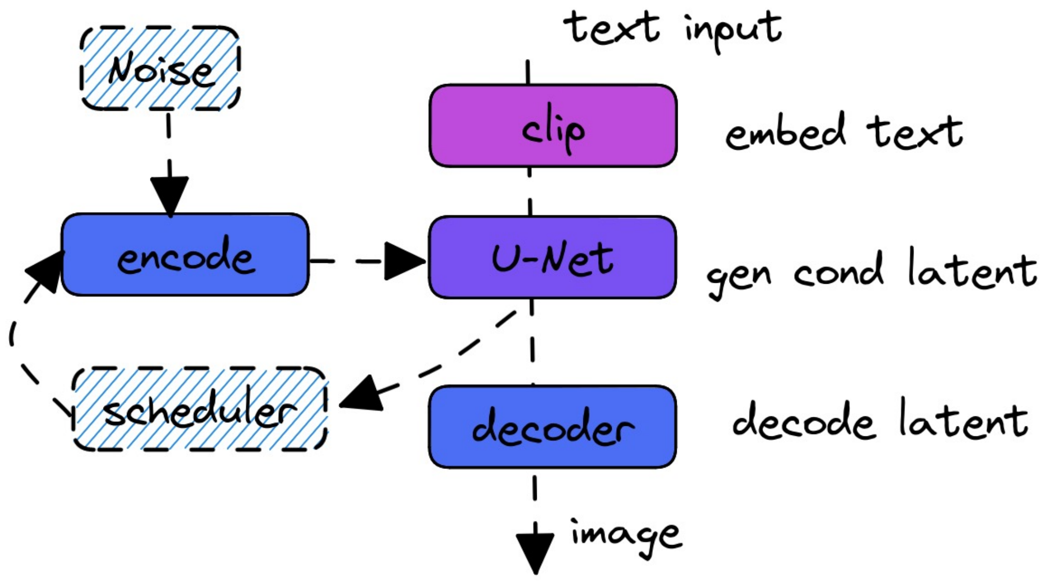
How to use *Intel Extension for PyTorch** with a complex model

Stable diffusion

The most basic form of stable diffusion is used to transform a text input to an image.

3 core components of a SD pipeline:

- **Text Encoding**: Transforms text into a high-dimensional distribution.
- **Diffusion**: Iteratively removes predicted noise (U-net) from a noisy image (VAE) conditioned on text encoding (CLIP) into an image distribution in the latent space.
- **Image Decoding**: Decodes image (VAE decoder) distribution into a coherent image.

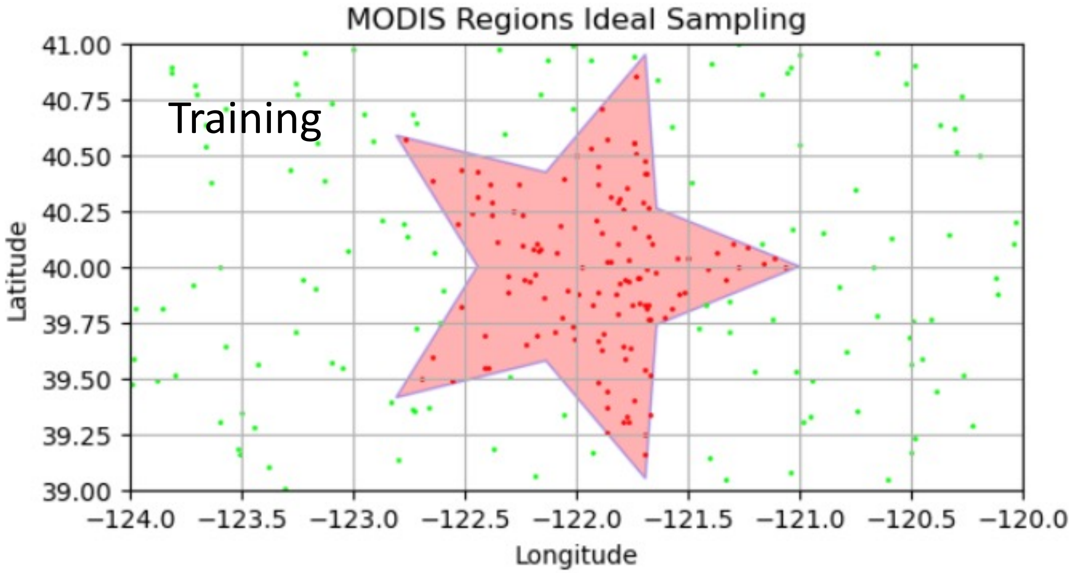


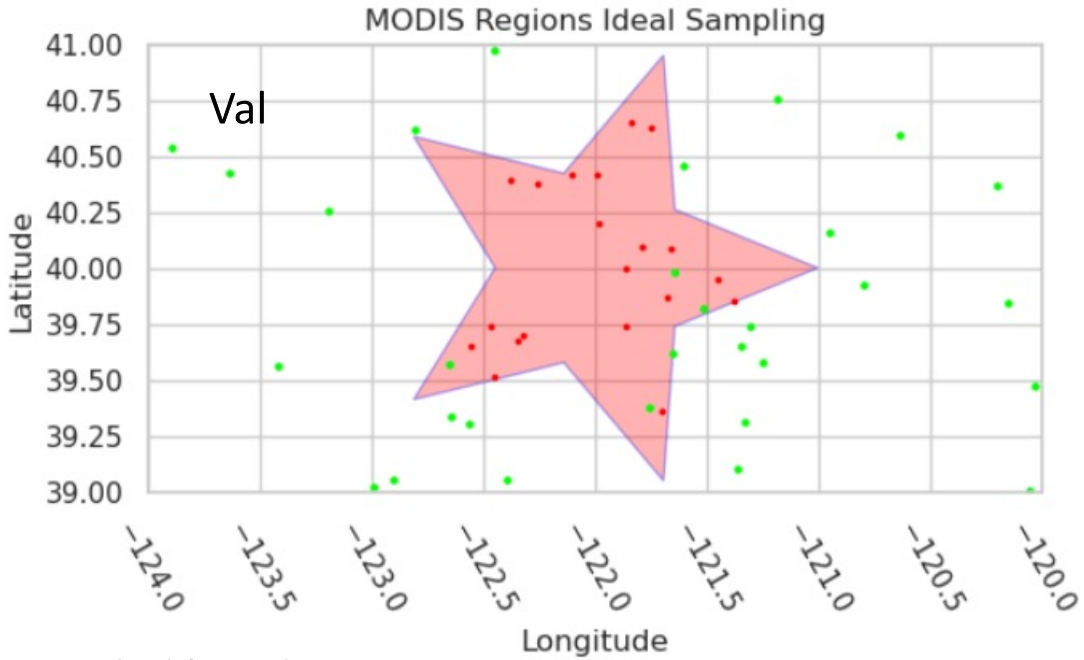
Optimizing SD using *Intel Extension for PyTorch* in 4 steps*

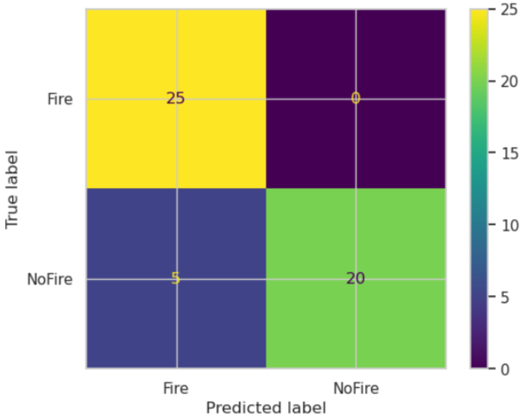
```python
def apply_ipex_optimize(pipe, dtype, input_example=None):
    pipe.unet = ipex.optimize(
        pipe.unet.eval(), dtype=dtype, inplace=True, sample_input=input_example
    )
    pipe.vae = ipex.optimize(pipe.vae.eval(), dtype=dtype, inplace=True)
    pipe.text_encoder = ipex.optimize(
        pipe.text_encoder.eval(), dtype=dtype, inplace=True
    )
```

- Instantiate a diffuser pipleline
- Pass to device (cpu vs xpu)
- Identify the subcomponents of the pipe
- Call ipex.optimize for each of the sub-components

Results







Can get as high an accuracy as needed by adjusting preparation step – RG values on synthetic set

Results – on Google* Earth Engine* map

- Apply model to all samples data not just test

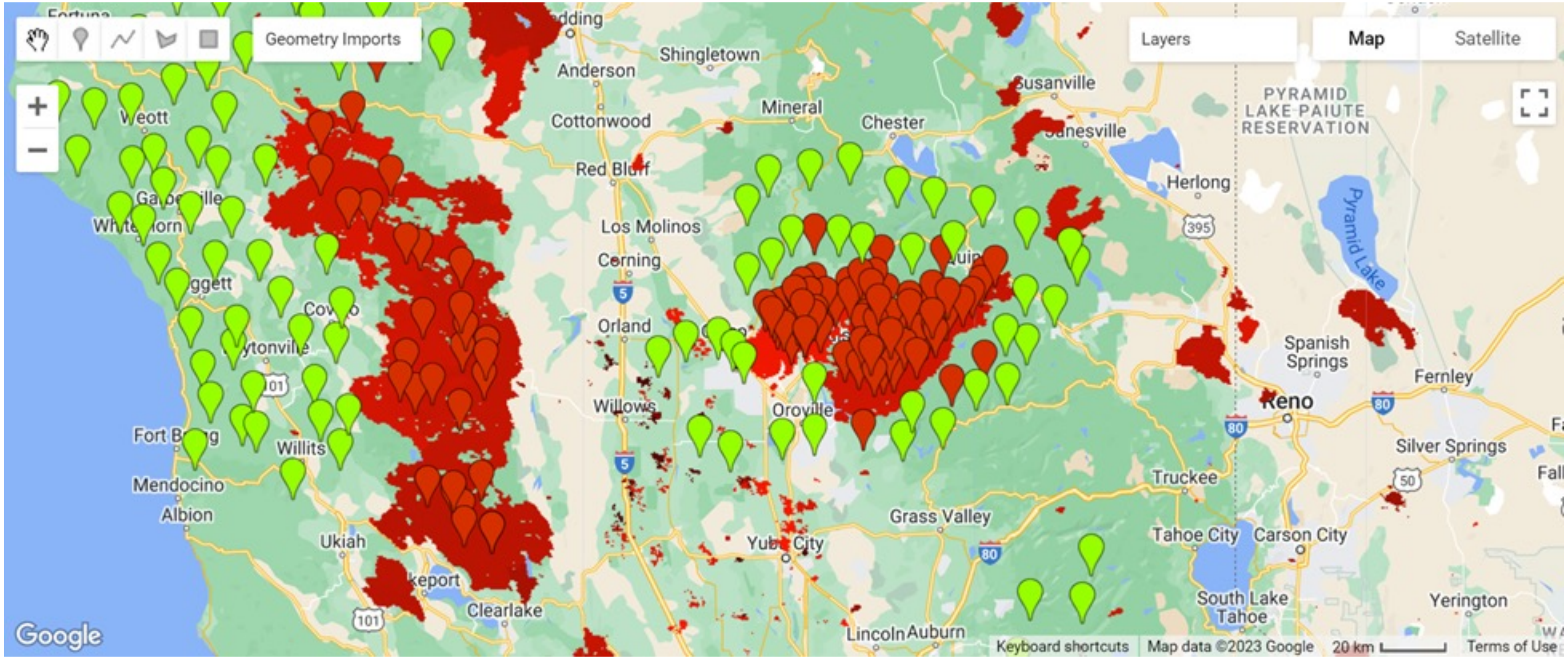


Figure 5. Showing inference from model on all train/ validation/ test data (red pins are predictions Fire=green pins, NoFire=red, the red polygon are regions of actual fire locations in 2018–2020.
Google Earth Engine with 'MODIS/006/MCD64A1' dataset

Call to Action!

- Use PyTorch for Geo-Spatial prediction!
- Optimize with Intel® Extension for PyTorch*
- Code for good! Help solve real world problems!

Try it yourself


Thank You
sythetic

Folder Structure