PyTorch* を使用して山火事を予測する

CPU で転移学習を使用して正確な画像分類器を効率良く開発

Bob Chesebrough インテル コーポレーション シニア AI ソリューション・アーキテクト

この記事では、PyTorch* による転移学習（英語）を使用して、画像のディテールのみを使用し、空中写真が伝える火災の危険性に従って空中写真を分類します。MODIS 火災データセット（英語）は、2018 年から 2020 年までにカリフォルニア州で発生した実際の火災を記録しています。MODIS（Moderate Resolution Imaging Spectroradiometer、中解像度イメージング分光放射計）データセットには、過去の山火事の場所の詳細な情報が得られるように、特定の日付範囲の高解像度画像とラベル付けされた地図データが含まれています。次に、火災が発生した地域および近隣地域の 2016 年から 2017 年の 2 年間の（火災が発生する前の）画像をサンプリングします。転移学習を使用して、「Fire（火災発生）」および「NoFire（火災なし）」とラベル付けされた数百枚の画像を追加し、（空中写真で事前トレーニングされていない）事前トレーニング済み ResNet 18 モデルを適応させます。

空中写真で使用するために（もともと ImageNet データセットでトレーニングされた）事前トレーニング済みモデルを微調整することは、山火事を予測するという状況下で画像から有用な情報を抽出する効果的なアプローチです。深い層とスキップ接続を備えた ResNet アーキテクチャーは、物体認識や画像分類を含むさまざまなコンピューター・ビジョン（英語）タスクで効果的であることが実証されています。このアプローチを使用すると、正確なモデルを構築するために必要なのは、数百枚の画像と約 15 分の CPU 時間だけです。詳細は、続きをお読みください。

空中写真を使用したケーススタディー

私のアプローチは、2016 年から 2021 年までの、カリフォルニア州の火災が発生した地域と発生していない地域の空中写真を利用した予測のみに焦点を当てたバイナリー分類器を作成することでした。トレーニング・セットには、2016 年から 2017 年までの、重要な地域の火災発生前の空中写真を使用しました。評価セットには、2018 年から 2020 年までに撮影された同じ場所の画像（および 2021 年の画像を含む拡張セット）を使用しました。火災が発生した地域と発生していない地域の両方をサンプリングしました。山火事の可能性は、MODIS データセットから取得した既知の山火事の地域に基づいています。サクラメントの北の、太平洋からシエラネバダ山脈までの地域を選択しました。この地域では、過去に大規模な山火事が何度も発生しています。

データの収集

データの取得と前処理は、次の基本的な手順に従いました。最初に、Google Earth Engine*（英語）と JavaScript* プログラムを使用して MODIS 火災データと空中写真を収集しました。プロジェクトのスクリプトは ForestFirePrediction（英語）リポジトリーにあります。次に、米国農務省の USDA/NAIP/DOQQ データセットから地図を生成しました。最後に、NASA MODIS/006/MCD64A1 データセットから空中写真を抜き出しました。

MODIS により定義された、2018 年から 2020 年までに火災が発生した場所と発生していない場所を**図 1** と**図 2** に示します。赤は火災が発生した地域です。オレンジと水色のピンは使用した画像のサンプルの場所で、オレンジのピンは火災が発生した場所を、水色のピンは火災が発生していない場所を表します。各画像は約 60 平方マイル（約 155 平方キロメートル）の範囲をカバーします。

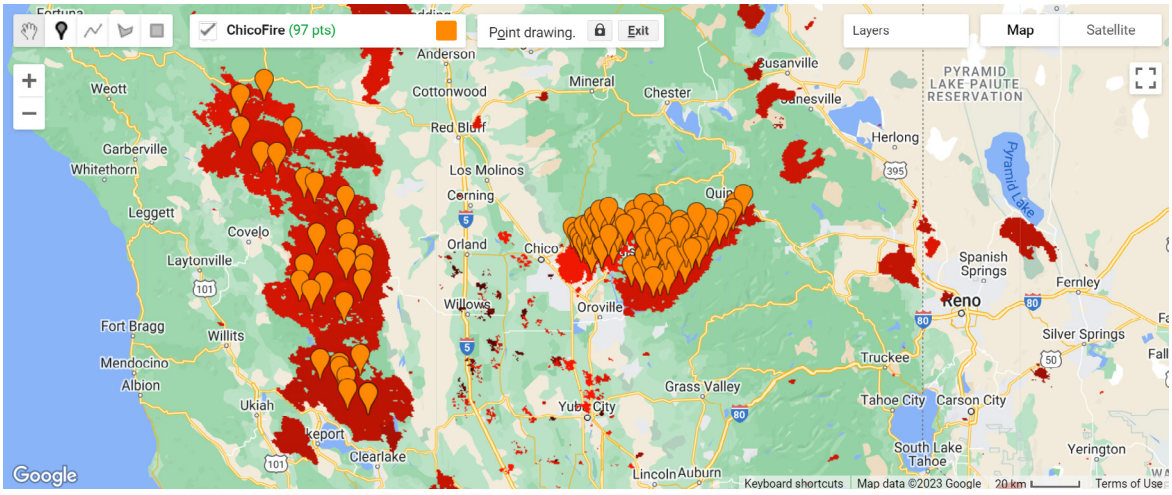

図 1. Google Earth Engine* と MODIS/006/MCD64A1 データセットを使用して山火事が発生した場所をサンプリング

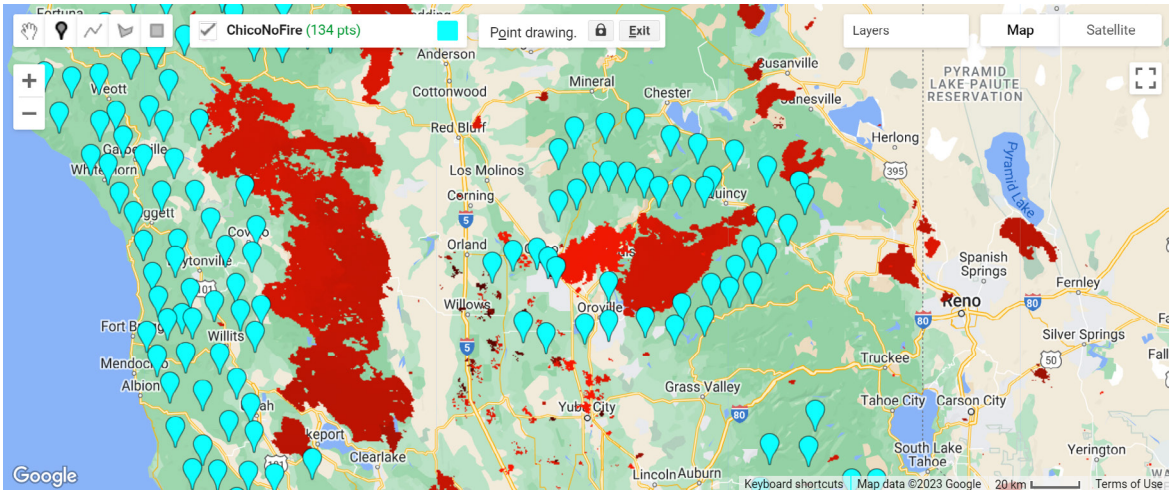

図 2. Google Earth Engine* と MODIS/006/MCD64A1 データセットを使用して山火事が発生していない場所をサンプリング

空中写真のサンプリングには、NAIP/DOQQ データセットを使用しました。例えば、**図 3** は、カリフォルニア州パラダイス付近の、大規模火災（2018 年）が発生する前の空中写真を示しています。

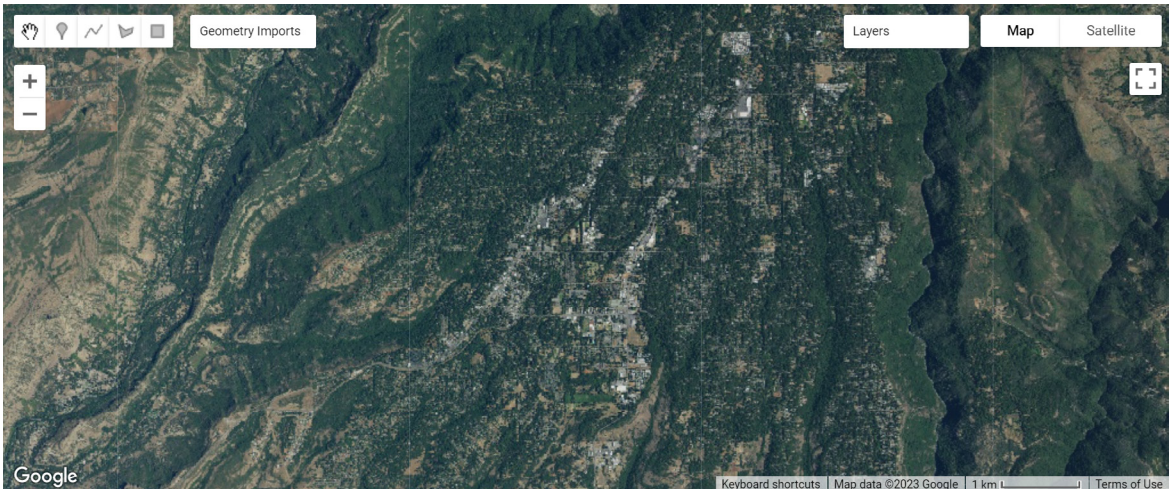

図 3. USDA/NAIP/DOQQ データセットの空中写真のサンプル (2018 年の山火事発生前のカリフォルニア州パラダイス)

火災が発生した地域のサンプルとして 106 枚の画像を、火災が発生していない地域のサンプルとして 111 枚の画像を使用しました（**表 1**）。転移学習を使用しているため、モデルを最初からトレーニングする場合よりもデータセットをはるかに小さくできます。画像のトレーニング、検証、テストの内訳は次のとおりです。

	トレーニング	検証	テスト
FIRE	87	9	10
NOFIRE	90	10	11

コード

モデルは ResNet-18 ベースです。ユーティリティー関数、トレーナークラス、モデルクラス、メトリクスクラスの 4 つの主要コードセクションを作成しました。さらに、最終的な混同行列とモデルの精度を表示するコードを追加しました。

インポート

```
import intel_extension_for_pytorch as ipex
import torch
import torch.nn as nn
import torchvision.models as models
from torch.utils.data import DataLoader
from torchvision import datasets, models, transforms
```

トレーニングと検証のデータセットを作成

トレーニング画像と検証画像の場所を定義し、各セット内の各画像の画像拡張を計算します。

```
num_physical_cores = psutil.cpu_count(logical=False)
data_dir = pathlib.Path("./data/output/")
TRAIN_DIR = data_dir / "train"
VALID_DIR = data_dir / "val"

...
```

トレーニングと検証セットのデータセット変換を定義

各画像で実行する一連の拡張を定義します。

```
...
img_transforms = {
    "train": transforms.Compose(
        [
            transforms.RandomHorizontalFlip(),
            transforms.RandomVerticalFlip(),
            transforms.RandomRotation(45),
            transforms.ToTensor(),
            transforms.Normalize(*imagenet_stats),
    ),
...
}
...
```

モデルのクラスを定義

モデルは、ResNet18 ベースのディープ・ニューラル・ネットワークのバイナリー分類器です。

```
class FireFinder(nn.Module):
...
    def __init__(self, backbone=18, simple=True, dropout= .4):
        super(FireFinder, self).__init__()
        backbones = {
            18: models.resnet18,
        }
            fc = nn.Sequential(
                nn.Linear(self.network.fc.in_features, 256),
                nn.ReLU(),
                nn.Dropout(dropout),
                nn.Linear(256, 2)
```

Trainer クラスを定義

このステップでは、インテル® AI アナリティクス・ツールキットの一部である、PyTorch 向けインテル® エクステンション（英語）を使用します。

```
class Trainer:
...
        self.loss_fn = torch.nn.CrossEntropyLoss()
        self.ipx = ipx
        self.epochs = epochs
        if isinstance(optimizer, torch.optim.Adam):
            self.lr = 2e-3
        self.optimizer = optimizer(self.model.parameters(), lr=lr)

    def train(self):
        self.model.train()
        t_epoch_loss, t_epoch_acc = 0.0, 0.0
        start = time.time()
        for inputs, labels in tqdm(train_dataloader, desc="tr loop"):
            inputs, labels = inputs.to(self.device), labels.to(self.device)
            if self.ipx:
                inputs = inputs.to(memory_format=torch.channels_last)
            self.optimizer.zero_grad()
            loss, acc = self.forward_pass(inputs, labels)
            loss.backward()
            self.optimizer.step()
            t_epoch_loss += loss.item()
            t_epoch_acc += acc.item()
        return (t_epoch_loss, t_epoch_acc)
...
    def _to_ipx(self):
        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
        )
```

モデルをトレーニング

20 エポック、ドロップアウト率 0.33、学習率 0.02 でモデルをトレーニングします。

```
epochs = 20
ipx = True
dropout = .33
lr = .02

torch.set_num_threads(num_physical_cores)
os.environ["KMP_AFFINITY"] = "granularity=fine,compact,1,0"

start = time.time()
model = FireFinder(simple=simple, dropout= dropout)
trainer = Trainer(model, lr = lr, epochs=epochs, ipx=ipx)
tft = trainer.fine_tune(train_dataloader, valid_dataloader)
```

モデルの推論

モデルに対して推論するクラスと関数を定義します。

```
class ImageFolderWithPaths(datasets.ImageFolder):
…
def infer(model, data_path: str):
    transform = transforms.Compose(
        [transforms.ToTensor(), transforms.Normalize(*imagenet_stats)]
    )
    data = ImageFolderWithPaths(data_path, transform=transform)
    dataloader = DataLoader(data, batch_size=4)
…
```

次のコード例は、モデルを使用して画像をスコアリングする方法を示しています。

```
images, yhats, img_paths = infer(model, data_path="./data//test/")
```

モデルの精度

次の混同行列は、21 のサンプル中に 2 つの誤検出があることを示しています。モデルの全体的な精度は約 89.9% です。

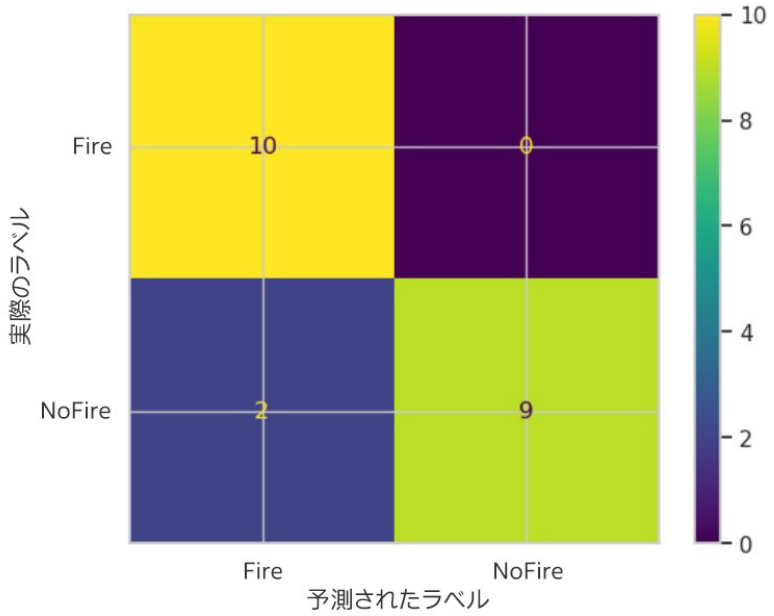

図 4 は、すべてのサンプル（トレーニング、検証、テスト）のモデルの予測を示しています。この図から、モデルが火災が発生する危険性のある地域を非常に正確に予測したことが分かります。緑のピンはモデルが火災が発生しないと予測したサンプルの場所で、赤のピンはモデルが火災が発生すると予想したサンプルの場所です。赤で表示されているのは、2018 年から 2020 年までに実際に火災が発生した地域です。

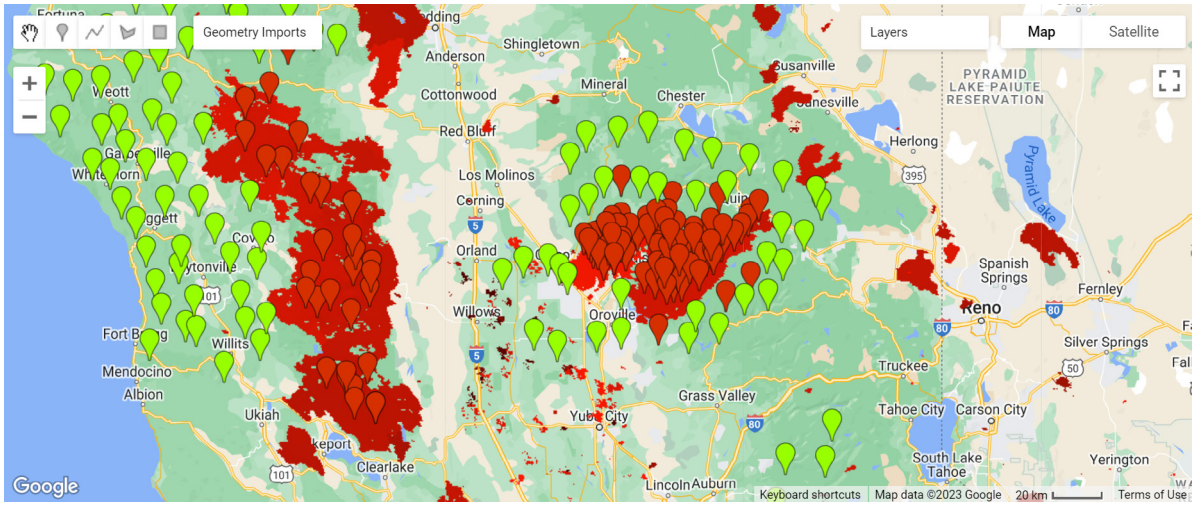

図 4. すべてのサンプルの推論結果

図 5 は、カリフォルニア州パラダイス付近の拡大図です。

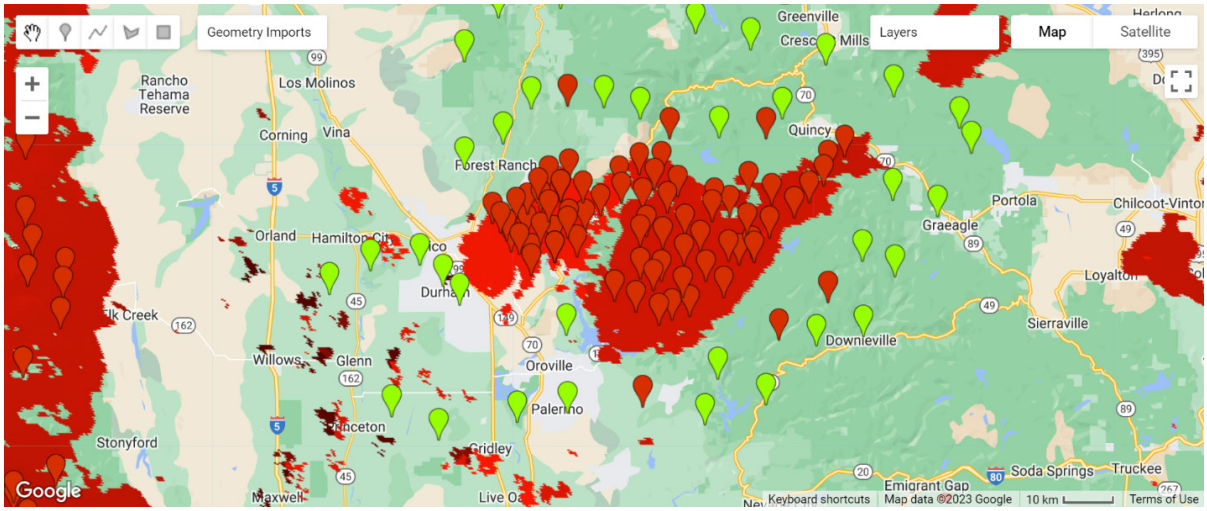

図 5. カリフォルニア州パラダイス付近のモデルの予測
(火災発生地域をほぼ正確に予測している)

まとめ

この記事では、PyTorch* が提供する画像解析機能とインテルの最適化機能を使用し、ResNet18 モデルをトレーニングおよびテストして、山火事を正確に予測する方法を説明しました。約 60 平方マイル(約 155 平方キロメートル)の範囲をカバーする空中写真をモデルに取り込むことにより、火災の有無を正確に予測できます。モデルの精度は現在約 89% ですが、反復回数を増やし、画像のセットを増やし、正則化に重点を置くことで、精度が向上する可能性があります。

この記事について議論したい方は、招待リンク（英語）をクリックしてインテル® DevHub Discord に参加してください。また、新しいインテル® デベロッパー・クラウド（英語）で、最新のインテルのハードウェア上でインテル® AI アナリティクス・ツールキットを試してみてください。