Fireflower detection

Javae Elliott and Joseph Strizhak

Problem Motivation

Air Patrols

Over vast amounts of suspected areas

Lookout Towers

Manned towers with extensive view

Local Reports

By general public, commercial & recreational pilots

Thermal Imaging

Detect hot spots & residual fires

Weather Monitoring

Lightning strikes, Moisture content, probability of wildfire

Executive Summary

Goal

Vision-based model to provide wildfire detection

Solution

Provide a CNN for object detection & localization

Benefit

Cost,Assistance to already-existing methods of detection

Technical Problems

Data Relevance

Most accessible were not specific to wildfires

Tensorflow Datasets

TFDS, AutoKeras, HParams, Tensorboard

Data Quality

Dataset size,image quality, curation

Performance

Model overfitting, metrics like f1 score ‘undoable’

Describe the feature and the current behavior/state.

Currently, F1-score cannot be meaningfully used as a metric in keras neural network models, because keras will call F1-score at each batch step at validation, which results in too small values. Therefore, F1-score was removed from keras, see keras-team/keras#5794, where also some quick solution is proposed. Tfa's F1-score exhibits exactly the same problem when used with keras.

Relevant information

Our Concept

In the same line of thinking, we thought of ways to get around the problem of fire image data quality.

Our Concept

Transfer learn a model trained on the tf_flowers ataset onto
our best curated* fire dataset

*removed non-natural objects, people. Kept only fire region of searched images. Balanced

Challenges & Background

HEAVILY
undocumented

TFDS

Use TFDS so as to
avoid RAM issues

AutoKeras
Tensorboard
HParams

Fire Dataset

1900 RGB images
950 per class

tf_flowers

3670 RGB images of
6 classes

[Dataset for Forest Fire Detection - Mendeley Data]   [tf_flowers | TensorFlow Datasets]

```python
clf = ak.ImageClassifier(
                         overwrite=True,
                         max_trials=10,
                         num_classes=5,
                         #tuner=>AutoKeras' Default tuner
                         loss='sparse_categorical_crossentropy',
                         directory='/content/',
                         seed=seed,
                         objective='val_accuracy'
                         #metric<-default accuracy
)

clf.fit(
        fl_train_data,
        batch_size=BATCH_SIZE,#put in to have autokeras change it if too much memory
        validation_split=0.2,
        epochs=5
)
print(clf.evaluate(fl_test_data))
```

```
Trial 10 Complete [00h 09m 28s]
val_accuracy: 0.9137324094772339

Best val_accuracy So Far: 0.9260563254356384
Total elapsed time: 01h 08m 22s
```

```python
def train_test_model(hparams):
  model = tf.keras.Sequential()
  for layer in autokeras_model.layers[:-2]: # Skip first and last layer
    model.add(layer)
  model.add(tf.keras.layers.Dropout(hparams[HP_DROPOUT]))
  model.add(tf.keras.layers.Dense(fire_num_classes, activation='softmax'))

  model.compile(
      optimizer = tf.keras.optimizers.Adam(learning_rate=hparams[HP_LR]),
      loss = tf.keras.losses.SparseCategoricalCrossentropy(),
      metrics=['accuracy'])

  history = model.fit(train_fire,
                      epochs=init_epochs)
  _,accuracy = model.evaluate(valid_fire)
  return accuracy

def run(run_dir, hparams):
  with tf.summary.create_file_writer(run_dir).as_default():
    hp.hparams(hparams)  # record the values used in this trial
    accuracy = train_test_model(hparams)
    tf.summary.scalar(METRIC_ACCURACY, accuracy, step=1)
```

Approach

We attempt a neural architecture search using AutoKeras on the tf_flowers dataset.

Then, using feature extraction, transfer learn that model onto the fire dataset, using Tensorboard & HParams to optimize performance.

Solution & Implementation

Hyperparameters

- [x] learning_rate

Min -infinity

Max +infinity

- [x] dropout

Min -infinity

Max +infinity

- [x] optimizer
  - [x] adam

Metrics

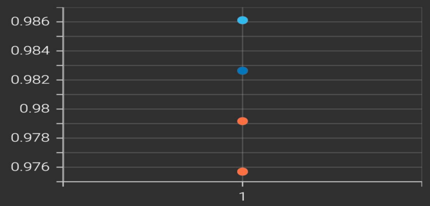
- [x] accuracy

Min -infinity Max +infinity

TABLE VIEW | PARALLEL COORDINATES VIEW | SCATTER PLOT MATRIX VIEW

Trial ID	Show Metrics	learning_rate	dropout	optimizer	accuracy
018c7e51f03f2...	☐	0.0064672	0.10000		1.0000
3338a144cd8b4...	☐	0.010000	0.20000		0.98611
39df7384dab67...	☐	0.010000	0.10000		0.98611
75ed77221f766...	☐	0.0064672	0.18260		0.97917
87e2e12931d42...	☐	0.0010000	0.19583		0.98611
99b73fa562c44...	☐	0.010000	0.10379		0.98958
9ca752c345ae4...	☐	0.0010000	0.10000	adam	0.97569
a17636c40bcf9...	☐	0.0010000	0.20000		0.97917
b138e04f2ec2c...	☐	0.0064672	0.20000		0.98611

accuracy
tag: accuracy



Name	Smoothed	Value	Step	Time
run-0	0.9757	0.9757	1	Sun Dec 18, 14:17:54
run-1	0.9826	0.9826	1	Sun Dec 18, 14:18:56
run-2	0.9826	0.9826	1	Sun Dec 18, 14:20:06
run-3	0.9861	0.9861	1	Sun Dec 18, 14:21:08
run-4	0.9861	0.9861	1	Sun Dec 18, 14:22:12
run-6	0.9757	0.9757	1	Sun Dec 18, 14:24:16
run-7	0.9792	0.9792	1	Sun Dec 18, 14:25:18
run-8	0.9792	0.9792	1	Sun Dec 18, 14:26:19

```
Layer (type)                 Output Shape              Param #
=================================================================
cast_to_float32 (CastToFloa  (None, 250, 250, 3)       0
t32)

normalization (Normalizatio  (None, 250, 250, 3)       7
n)

random_flip (RandomFlip)     (None, 250, 250, 3)       0

random_rotation (RandomRota  (None, 250, 250, 3)       0
tion)

efficientnetb7 (Functional)  (None, None, None, 2560)  64097687

global_average_pooling2d (G  (None, 2560)              0
lobalAveragePooling2D)

dropout_2 (Dropout)          (None, 2560)              0

dense_1 (Dense)              (None, 2)                 5122

=================================================================
Total params: 64,102,816
Trainable params: 5,122
Non-trainable params: 64,097,694
```

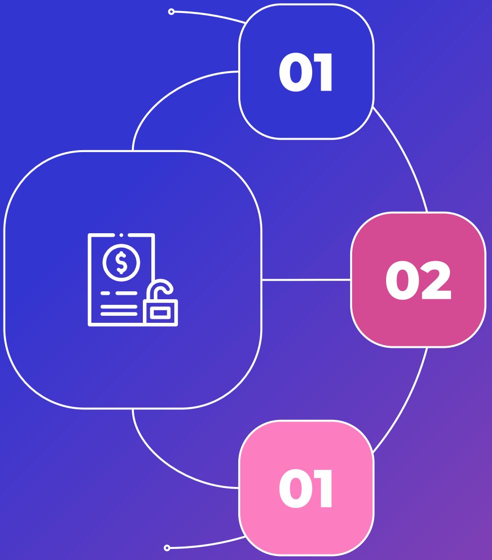
AutoKeras found a model (EfficientNet pre-trained on ImageNet) that yielded top validation accuracy.

Tensorboard found minimal gain in feature extraction between dropout & learning rate.

We end with (on test set)

`Accuracy: 0.955263 Loss: 0.128732`

Localization



Transfer learning

- Pre-trained model from Tensorflow Object Detection API

Bounding boxes for fires

- Used label studio for picture labeling

Re-training/fine-tuning

- Retrained the Tensorflow model for fire detection

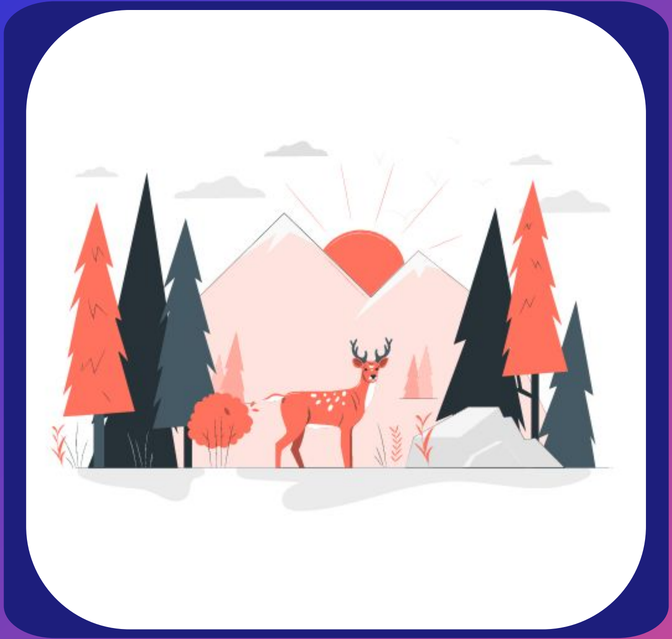
Possibilities & Improvements

Video, not Image

Object detection models trained on videos are more robust

BoundBox Space

Some images are engulfed in flame, its effect on localization would be harmful



Tensorboard

Tensorboard has a lot more visualizational range that would've been ideal to explore

Precision

Due to API issues, we can't exactly measure some relevant metrics for our classifier.

Thanks!

Javae Elliott and Joseph Strizhak

GitHub Link

Fireflower detection

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**