Transfer Learning for Fire Detection: Tackling the FLAME DataSet

Jayani Dipalkumar Bhatwadiya
Department Of Mathematical Science
University of Essex
Colchester, UNITED KINGDOM
Registration Number : 2004351

***Abstract*—In recent advances the Transfer Learning technique is proven to best image classification approach due to the reason of reusability of pre-trained model. In this study one of the best transfer learning technique is used to detect fire from the given IEEE dataset which having binary classes such as "Fire" and "No-Fire". EfficientNet model consist of new baseline network which uses the AutoML MNAS framework,gives both good accuracy and efficiency. Image Augmentation method used to reshape, flip, roate the images and then passed to the network then model is compiled thorough "Adam" optimizer which gives better performance.**

***Index Terms*—Convolutional neural networks, EfficientNet**

I. Introduction

WildFires are natural disaster that damage environmental areas like forests, grassland, homes etc. There are many reasons to occur WildFires which includes Human Activities, Volcano Eruption, Heat Waves and Climate Change. From the past decades it has been viewed that there are several damages occured through WildFire. Firstly, in the city of California "the 2018 Camp Fire" damaged the whole city and died almost 86 people. Secondly, Bush Fires in Australia are the worst which breaks history records of decades.Millions of forest have burned, and created high level of air pollution. In this fire around 90,000 people had to move to the other area and billions of Koalas and Kangaroos have died.

Traditional approaches to find the fires in the reflected area to look from Helicopters, Air-crafts and Tallest buildings. In Recent Research IOT (Internet of Things) Machines has developed to detect the fire. Its based on wireless sensor network, but these type of IOT devices requires further investigation and testing before use, and it requires highly investment also.[11]

Considering the challenges and issues of these methods, in Recent decade there are Artificial Intelligence(AI) and Machine Learning based image classification models is used to analysis fire detection, which is proven even more successful than IOT Devices and Traditional methods. The Boon from NanoTechnology, there are semiconductors of new generation which are "Tensor Processing Units (TPUs)" and "Graphical Processing Units (GPUs)" provides faster performance.[11]

In this study, The Binary image classification was carried out of dataset IEEE dataset of FLAME (Fire Luminosity Airborne based Machine learning Evaluation).It has two classes with "Fire" and "No-Fire" from Training and Test Dataset. The inital Approach EffiencientNet model is used to classify the images since it proofs to be best image Classification model in Transfer Learning which gives high accuracy.The rest of study consists of Literature Review, Methodolgy, Results, Discussions and Conclusion.

https://github.com/Jayani/CE888-Assignment-2

II. Literature Review

In this section some approaches and methods highlighted to early detect fire. The Early Fire-Alarm method based on video processing to detect fire.It uses RGB (red, green, blue) model to detect smoke pixels from the fire images, when the smoke pixels met with fire alarm then the alarm will ran when the rising condition is met, This system is used by important military, social security, commercial applications [2] Texture is important to detect Fire smoke. After combining two texture analysis tool such as Wavelet Analysis and Gray Level Cooccurrence Matrices (GLCM). It converts into extractable features to detect fire smoke, This perfromace is associated with how many number of input vectors passed.[4] The other fire alarming system based on video processing in which the smoke pixel judgement done by two rules : a chromaticity-based static decision rule and a diffusion-based dynamic characteristic decision rule.[3] Fine-Tuned convolutional neural networks cameras used to detect fire indoor and outdoor. Dynamic Channel selection algorithm is used to detect the fire which based on radio networks.[10] To detect certain type of fire, smoke and any explosions, one technique is used which is based on cascaded approach through camera monitoring. The result of this system showed higher fire detection rate up to 95%. [8] Unmanned Aerial Vehicles (UAVs), Fire detection platfrom uses computer vision techniques to detect smoke or fire.[6] A Cyber-Physical Social System (CPSS) which uses three-level CPSS to detect early fire detction. It uses IOT approach to identify emergency situations.[1] To detect smoke gas sensor is also used which is highly sensitive and gives rapid response based on metal oxides.[7]

III. Methodology

A. Transfer Learning

Transfer Learning is a Machine Learning method in which model is trained and developed for one task and that model can be again use by another task, which is related to that first

task. [5] Its a learning process in which one model information will transfer to the another model. It is best approach to train any neural network model to increase speed and performance on the huge and challenging data.

Its widely used technique to deal with image dataset, because images contain high resolution and pixels, so transfer learning plays good role to get higher performance on the trained model. There are many examples of pre-trained model,uses transfer learning method to train new data are Inception, DenseNet, NASNet, VGG, ResNet, AlexNet, Efficientnet. Among all these pre-trained models "EfficientNets" gives 10x better efficiency In this paper "EfficientNets" model is used to train the given dataset.

B. EfficientNets Model

EfficientNets is a model obtain from Convolutional neural networks (CNNs). Its reduced the number of parameters and provide high perfromance compared to other models of (CNNs).

1) EfficientNets Architecture: The model's effectiveness is highly depend on baseline network of the model. To improve performance, new baseline network was added for performing neural architecture search, which use AutoML MNAS Framework,optimizes accuracy and efficiency (FLOPS). The result architecture uses (MBConv), which is called "Mobile Inverted Bottleneck Convolution". It is similar architecture with MobileNetV2 and MnasNet, then after scale up the baseline network to achieve the architecture of "EfficientNets". (Fig. 1.) shows the architecture of Efficient Net model. [9]

2) EfficientNet Performance: In the Comparison with the Convolutional Neural Network, both higher accuracy and better efficiency achieved by the EfficientNet models. Its proven that it is having good performance on image dataset, that there are exampels like CIFAR-100 datasets with (91.7%) and Flowers dataset with (98.8%) accuracy.

(Fig. 2.) shows the comparison of other transfer learning model with EfficientNet model.

C. The Fire Image Dataset

The Aerial Pile Burn Detection Dataset (The FLAME Dataset) consists total of 47,992 images which is divided in "Training" and "Test". Its further split into two disturbance classes called "Fire" and "No-Fire".Both Training and Test data contains 39,375 and 8617 images respectively. The images size is 254x254 pixels of both "Training" and "Test" data and all are in ".jpg" format. This dataset is available on IEEE-Dataport, the whole dataset siez is 1.46 GB which is in zip format, and the images taken using drones during a prescribed pile burn in Northern Arizona, USA.

D. Exploratory Data Analysis

After uploading the data in "Kaggle", The count of images which belongs from 2 different data "Training" and "Test" were calculated, and separated "Training" images in Train/Validation split by applying 0.20. After applying validation split the Training and Validation images are 31501 and 7874 respectively. Images from two different classes "Fire" and "No-Fire" has been calculated and viewed 6 random images from these classes. Used "matplotlib" library to view graphical representation of the data and plotted Training and Test data distribution. EffiecientNet model consists of loads weights that used for pre-trained on ImageNet, the first fully-connected layer at the top of the network which is optional, other is "GlobalAveragePooling2D" which treats the layer as feature extractions. The whole model summary is shown in (Fig. 3.) The model is trained with the loss of "binary crossentropy" because the dataset consists of two classes. Optimizer named "Adam" is used with learning rate of 0.001

Fig. 1. EfficientNet Model Architecture

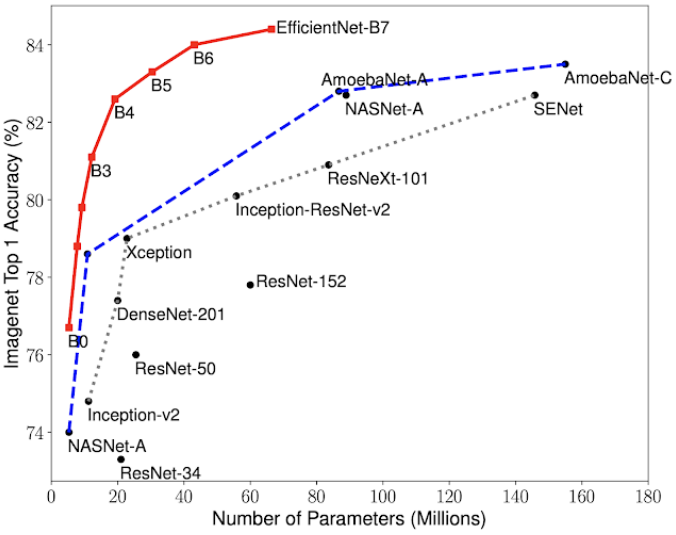

Fig. 2. EfficientNet Performance

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
efficientnet-b0 (Functional) (None, 7, 7, 1280)        4049564
_________________________________________________________________
global_average_pooling2d_3 ( (None, 1280)              0
_________________________________________________________________
dropout_3 (Dropout)          (None, 1280)              0
_________________________________________________________________
dense_7 (Dense)              (None, 2)                 2562
=================================================================
Total params: 4,052,126
Trainable params: 4,010,110
Non-trainable params: 42,016
_________________________________________________________________
```

Fig. 3. Model Summary

to monitor Training and Validation accuracy.

E. Data Pre-Processing

Data Generators are used to read the images from Training Dataset. It converts the images to tensors and pass them to the network model. To convert images to tensors "ImageDataGenerator" is used, in which images are normalised from zero to one range. Images are resized with (IMAGE-WIDTH, IMAGE-HEIGHT) = (224*224) and the BATCH SIZE of 32. These operation applied to both Training Generator with training images and Validation Generator with validation images.

1) Image Augmentation: Image Augmentation is a technique to pre-process the training images in multiple variations of images to avoid overfitting/underfitting of data and increase the accuracy. The common techniques are used to generate new images are:

1) Flip horizontally or vertically
2) Rotate at some degrees
3) Scaling outward or inward
4) Crop randomly
5) Translate the images

```
# ImageDataGenerator for training pictures with data augmentation
train_datagen = ImageDataGenerator (
            preprocessing_function=preprocess_input,
            rescale=1./255,
            horizontal_flip = True,
            validation_split = 0.2, # Split into Train/Validation
            featurewise_center = False,  # set input mean to 0 over the dataset
            samplewise_center = False,  # set each sample mean to 0
            featurewise_std_normalization = False,  # divide inputs by std of the dataset
            samplewise_std_normalization = False,  # divide each input by its std
            zca_whitening = False,  # apply ZCA whitening
            rotation_range = 40,  # randomly rotate images in the range (degrees, 0 to 180)
            zoom_range = 0.2, # Randomly zoom image
            width_shift_range=0.2,  # randomly shift images horizontally (fraction of total width)
            height_shift_range = 0.2,  # randomly shift images vertically (fraction of total height)
            vertical_flip = False,  # randomly flip images
            shear_range=0.2,
            fill_mode='nearest',)
```

Fig. 4. Image Augmentation

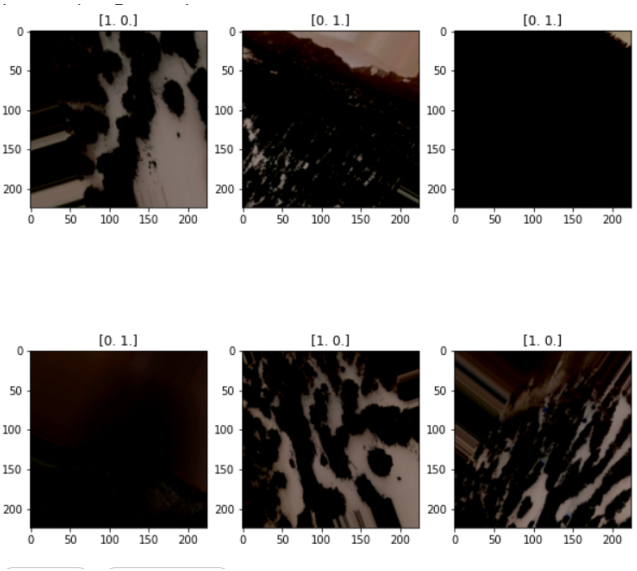

Fig. 5. Augmented Images

ImageDataGenerator API from Keras was used to augment Training images. The basic code to create ImageDataGenerator is shown in the (Fig. 4.)

(Fig. 5) shows the augmented images on which some operation such as Flip, Rotation, Scaling are performed.

F. Training

The model is trained with Training and Validation data which was augmented through "ImageDataGenerator" with 10 epochs, 34 steps per epoch. All Training and Validation images are used to train the model.

```
Epoch 1/10
984/984 [==============================] - 652s 654ms/step - loss: 0.0648 - binary_accuracy: 0.9762 - val_loss: 0.0308 - val_binary_accuracy: 0.9892
Epoch 2/10
984/984 [==============================] - 563s 572ms/step - loss: 0.0199 - binary_accuracy: 0.9933 - val_loss: 0.2081 - val_binary_accuracy: 0.9333
Epoch 3/10
984/984 [==============================] - 573s 582ms/step - loss: 0.0192 - binary_accuracy: 0.9931 - val_loss: 0.0518 - val_binary_accuracy: 0.9850
Epoch 4/10
984/984 [==============================] - 582s 591ms/step - loss: 0.0174 - binary_accuracy: 0.9936 - val_loss: 0.0424 - val_binary_accuracy: 0.9879
Epoch 5/10
984/984 [==============================] - 545s 554ms/step - loss: 0.0154 - binary_accuracy: 0.9948 - val_loss: 0.1611 - val_binary_accuracy: 0.9640
Epoch 6/10
984/984 [==============================] - 542s 550ms/step - loss: 0.0140 - binary_accuracy: 0.9951 - val_loss: 0.0352 - val_binary_accuracy: 0.9893

Epoch 00006: ReduceLROnPlateau reducing learning rate to 0.00050000002374872577.
Epoch 7/10
984/984 [==============================] - 556s 565ms/step - loss: 0.0098 - binary_accuracy: 0.9970 - val_loss: 0.0909 - val_binary_accuracy: 0.9745
Epoch 8/10
984/984 [==============================] - 550s 559ms/step - loss: 0.0089 - binary_accuracy: 0.9966 - val_loss: 0.0912 - val_binary_accuracy: 0.9768
Epoch 9/10
984/984 [==============================] - 542s 550ms/step - loss: 0.0074 - binary_accuracy: 0.9973 - val_loss: 0.0784 - val_binary_accuracy: 0.9780
Epoch 10/10
984/984 [==============================] - 543s 552ms/step - loss: 0.0102 - binary_accuracy: 0.9967 - val_loss: 0.0776 - val_binary_accuracy: 0.9803
```

Fig. 6. Example of a figure caption.

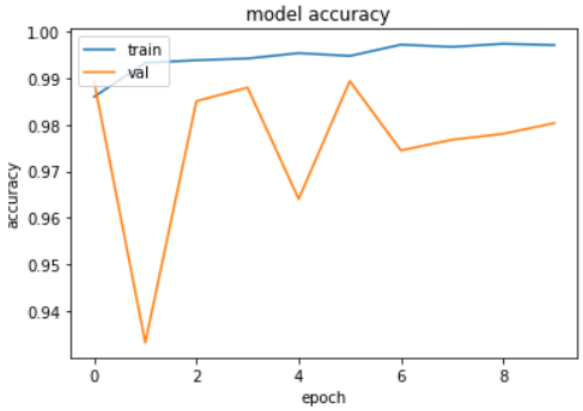

Fig. 7. Training/Validation Accuracy

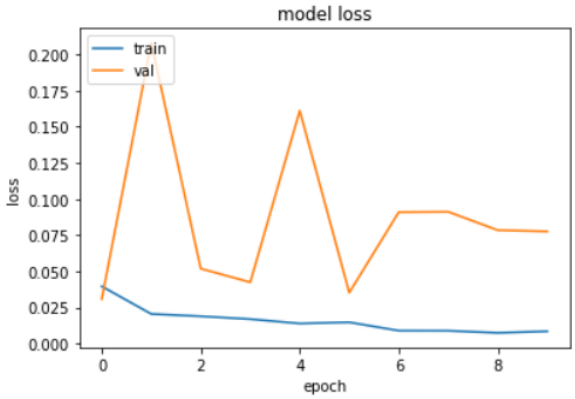

Fig. 8. Training/Validation Loss

IV. Results

All the Training, Validation and Testing data are performed in Kaggle's inbult GPU. Kaggle provides free access to Nvidia K80 GPUs in kernels, which speedup 12.5X faster during training of any Neural Network Model. [11]

The Training Dataset is divided into 80% of Training Images and 20% Validation Images. Thus while training the model, in the Training section 25,018 images of class "Fire" and 14,357 images of class "No-Fire" included from total number of 39,375 Training images. All images are shuffled and augmented such as flip, normalise, rotated before passing into the network. The Training of the model ran over 10 epochs with the Adam Optimizer which is set to 0.001, and loss of binary crossentropy, because it is having two classes "Fire" and "No-Fire" with the BATCH SIZE of 32. To evaluate accuracy of Test data, from total number of 8617 Testing images, includes 5137 "Fire" images and 3480 "No-Fire" images passed into pre-trained model. Given table shows loss and accuracy on Training, Validation and Testing Dataset.[11]

DataSet	Loss	Accuracy(%)
Training Set	0.0056	99.77
Validation Set	0.0785	98.06
Test Set	0.7618	75.78

The Loss and Accuracy of Training and Validation sets is shown in (Fig. 7 Fig. 8) respectively.

V. Discussion

The model can be shown as Overfitting (Fig. 7.). The Blue Line is representing Training Accuracy which is very closer to 100 percentage but the Validation Accuracy which is having Yellow Line is very low. The difference between this two is very vast. Thus We need to build very complex model, may be it can have extract more features.

As the model is Overfitting, we need to generalise the data more. By increasing number of of Conv2D layers and dropout if necessary make model more deeper to train the data.

This experiments done with the other models like CNN, VGG16 and ResNet50. The result will be varies on some criteria such as input Layers, image width and image height and batch size. The result of these three models are 52%, 59% and 67% respectively. The best result reported through existing Efficient Net model which is 75.78% in Test Dataset. Therefore, I used Efficient Net architecture for this purpose. However, some improvement still needed to avoid overfitting.

VI. Conclusion

This study gone through the FLAME (Fire Luminosity Airborne-based Machine learning Evaluation) dataset. The transfer learning method is used to achieve best result. By training Efficient Net model we got Training 99.77%, Validation 98.06% and Testing 75.78% accordingly, which is actually very good.

However, By implementing of Fine Tuning and K-Fold Cross validation we can achieve best result. Fine Tuning consists of freezing the layer of the model then train the model then finally unfreeze whole model and again train it on the new data at low percentage. K-Fold Cross Validation technique is use to split the data in train and test in equal size set and we have to define number of folds that will increase and iterate over the folds.

References

[1] Marios Avgeris, Dimitrios Spatharakis, Dimitrios Dechouniotis, Nikos Kalatzis, Ioanna Roussaki, and Symeon Papavassiliou. Where there is fire there is smoke: A scalable edge computing framework for early fire detection. *Sensors*, 19(3), 2019.

[2] Thou-Ho Chen, Ping-Hsueh Wu, and Yung-Chuen Chiou. An early fire-detection method based on image processing. In *2004 International Conference on Image Processing, 2004. ICIP '04.*, volume 3, pages 1707–1710 Vol. 3, 2004.

[3] Thou-Ho Chen, Yen-Hui Yin, Shi-Feng Huang, and Yan-Ting Ye. The smoke detection for early fire-alarming system base on video processing. In *2006 International Conference on Intelligent Information Hiding and Multimedia*, pages 427–430, 2006.

[4] Yu Cui, Hua Dong, and Enze Zhou. An early fire detection method based on smoke texture analysis and discrimination. In *2008 Congress on Image and Signal Processing*, volume 3, pages 95–99, 2008.

[5] Mahbub Hussain, Jordan Bird, and Diego Faria. A study on cnn transfer learning for image classification. 06 2018.

[6] Diyana Kinaneva, Georgi Hristov, Jordan Raychev, and Plamen Zahariev. Early forest fire detection using drones and artificial intelligence. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1060–1065, 2019.

[7] Kwangjae Lee, Young-Seok Shim, Young Geun Song, Soo Deok Han, Youn-Sung Lee, and Chong-Yun Kang. Highly sensitive sensors based on metal-oxide nanocolumns for fire detection. *Sensors*, 17(2), 2017.

[8] Oleksii Maksymiv, Taras Rak, and Dmytro Peleshko. Real-time fire detection method combining adaboost, lbp and convolutional neural network in video sequence. In *2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*, pages 351–353, 2017.

[9] Staff Software Engineer Mingxing Tan and Google AI Quoc V. Le, Principal Scientist. Efficientnet: Improving accuracy and efficiency through automl and model scaling, 2019.

[10] Khan Muhammad, Jamil Ahmad, and Sung Wook Baik. Early fire detection using convolutional neural networks during surveillance for effective disaster management. *Neurocomputing*, 288:30–42, 2018. Learning System in Real-time Machine Vision.

[11] Alireza Shamsoshoara, Fatemeh Afghah, Abolfazl Razi, Liming Zheng, Peter Z. Fulé, and Erik Blasch. Aerial imagery pile burn detection using deep learning: The flame dataset. *Computer Networks*, 193:108001, 2021.