# Experimentally Defined Convolutional Neural Network Architecture Variants for Non-temporal Real-time Fire Detection

[and subsequent follow on work: \_Experimental Exploration of Compact Convolutional Neural Network Architectures

for Non-temporal Real-time Fire Detection\_]

![Python Build/Test](https://github.com/tobybreckon/fire-detection-cnn/workflows/Python%20Build/Test/badge.svg) Tested using Python 3.7.x, [TensorFlow 1.15](https://www.tensorflow.org/install/), [TFLearn 0.3.2](http://tflearn.org/) and [OpenCV 3.x / 4.x](http://www.opencv.org) (requires opencv extra modules - ximgproc module for superpixel segmentation)

## Architectures:

![FireNet](https://github.com/tobybreckon/fire-detection-cnn/blob/master/images/FireNet.png)

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;FireNet architecture (above)

![InceptionV1-onFire](https://github.com/tobybreckon/fire-detection-cnn/blob/master/images/InceptionV1-OnFire.png)

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;InceptionV1-OnFire architecture (above)

![InceptionV3-onFire](https://github.com/tobybreckon/fire-detection-cnn/blob/master/images/InceptionV3-OnFire.png)

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;InceptionV3-OnFire architecture (above)

![InceptionV4-onFire](https://github.com/tobybreckon/fire-detection-cnn/blob/master/images/InceptionV4-OnFire.png)

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;InceptionV4-OnFire architecture (above)

## Abstract:

\_"In this work we investigate the automatic detection of fire pixel regions in video (or still) imagery within real-time

bounds without reliance on temporal scene information. As an extension to prior work in the field, we consider the performance of experimentally defined, reduced complexity deep convolutional neural network (CNN) architectures for this task. Contrary to contemporary trends in the field, our work illustrates

maximal accuracy of 0.93 for whole image binary fire detection (1), with 0.89 accuracy within our superpixel localization

framework can be achieved (2), via a network architecture of significantly reduced complexity. These reduced architectures

additionally offer a 3-4 fold increase in computational performance offering up to 17 fps processing on contemporary

hardware independent of temporal information (1). We show the relative performance achieved against prior work using

benchmark datasets to illustrate maximally robust real-time fire region detection."\_

(1) using InceptionV1-OnFire CNN model (2) using SP-InceptionV1-OnFire CNN model

[[Dunnings, Breckon, In Proc. International Conference on Image Processing, IEEE, 2018](https://breckon.org/toby/publications/papers/dunnings18fire.pdf)]

\_".... Contrary to contemporary trends in the field, our work illustrates a maximum overall accuracy of 0.96 for full frame binary fire detection (3) and 0.94 for superpixel localization (4) using an experimentally defined reduced CNN architecture based on the concept of InceptionV4. We notably achieve a lower false positive rate of 0.06 compared to prior work in the field presenting an efficient, robust and real-time solution for fire region detection."\_

(3) using InceptionV4-OnFire CNN model (4) using SP-InceptionV4-OnFire CNN model

[[Samarth, Bhowmik, Breckon, In Proc. International Conference on Machine Learning Applications, IEEE, 2019](https://breckon.org/toby/publications/papers/samarth19fire.pdf)]

---

## Reference implementation:

Put simply, our full-frame binary detection (\_FireNet, InceptionV1-OnFire, InceptionV3-OnFire, InceptionV4-OnFire\_) architectures determine whether an image frame contains fire globally, whereas the superpixel based approaches (\_SP-InceptionV1-OnFire, SP-InceptionV3-OnFire, SP-InceptionV4-OnFire\_) breaks down the frame into segments and performs classification on each superpixel segment to provide in-frame localization.

This respository contains the ```firenet.py``` and ```inceptionVxOnFire.py``` files corresponding to the binary (full-frame) detection models from the paper(s). In addition the ```superpixel-inceptionVxOnFire.py``` file corresponds to the superpixel based in-frame fire localization from the paper(s).

To use these scripts the pre-trained network models must be downloaded using the shell script ```download-models.sh``` which will create an additional ```models``` directory containing the network weight data (on Linux/MacOS). Alternatively, you can manually download the pre-trained network models from [http://dx.doi.org/10.15128/r19880vq98m](http://dx.doi.org/10.15128/r19880vq98m) [Dunnings, 2018] + [http://doi.org/10.15128/r25x21tf409](http://doi.org/10.15128/r25x21tf409) [Samarth, 2019] and unzip them to a directory called ```models``` in the same place as the python files.

The superpixel based approach was trained to perform superpixel based fire detection and localization within a given frame as follows:

\* image frame is split into segments using SLIC superpixel segmentation

\* a \_SP-InceptionVx-OnFire\_ convolutional architecture (for \_x = 1, 3, 4 for InceptionV1, InceptionV3, InceptionV4\_), trained to detect fire in a given superpixel segment, is used on each superpixel.

\* at run-time (inference), the selected \_SP-InceptionVx-OnFire\_, network is run on every superpixel from the SLIC segmentation of the image

#### \_Which model should I use ?\_

For the \*\*best detection performance (i.e. high true positive rate, low false positive rate) use \_InceptionV4-OnFire\_\*\* (example: ```inceptionVxOnFire.py -m 4```) which operates at 12 frames per second (fps), for \*\*best throughtput (17 fps) use \_FireNet\_\*\* (example: ```firenet.py```) which has slightly lesser performance (i.e. lower true positive rate, higher false positive rate).

The \_InceptionV1-OnFire\_ and \_InceptionV3-OnFire\_ offer alternative performance characteristics in terms of detection, false alarm and throughput - (example: ```inceptionVxOnFire.py -m 1``` or ```inceptionVxOnFire.py -m 3```).

The \*\*\_SP-InceptionV4-OnFire\_ model offers the best superpixel localization detection performance\*\* of the fire within the image (example: ```superpixel-inceptionVxOnFire.py -m 4```) but at a lower throughput than the alternative, lesser detection performing \_SP-InceptionV1-OnFire\_ and \_SP-InceptionV3-OnFire\_ superpixel models (example: ```superpixel-inceptionVxOnFire.py -m 1``` or ```superpixel-inceptionVxOnFire.py -m 3```). For full comparison see most recent paper - [[Samath, 2019](https://breckon.org/toby/publications/papers/samarth19fire.pdf)]

\_Caveat:\_ \*\*if you need to convert the models\*\* to protocol buffer (.pb) format (used by [OpenCV](http://www.opencv.org) DNN, [TensorFlow](https://www.tensorflow.org/), ...) and also tflite (used with [TensorFlow](https://www.tensorflow.org/)) then use the \*\*\_FireNet\_\*\* or \*\*\_InceptionV1-OnFire\_ / \_SP-InceptionV1-OnFire\_\*\* versions at the moment as, due to a long-standing [issue](https://github.com/tensorflow/tensorflow/issues/3628) in TensorFlow with the export (freezing) of the Batch Normalization layers, the protocol buffer (.pb) format and .tflite versions of the \_...V3-OnFire\_ and \_...V4-OnFire\_ have significantly lesser performance (which is to be expected, given the approach we use to workaround the problem).

---

## Fire Detection Datasets:

The custom dataset used for training and evaluation can be found on [[Durham Collections - Dunnings/Breckon, 2018](https://collections.durham.ac.uk/collections/r1ww72bb497)] and [[Durham Collections - Samarth/Breckon, 2019](https://collections.durham.ac.uk/collections/r2jm214p16f)] (together with the trained network models). A direct download link for the dataset is [[Dunnings, 2018 - original data](https://collections.durham.ac.uk/downloads/r2d217qp536)] and [[Samarth, 2019 - additional data](https://collections.durham.ac.uk/downloads/r10r967374q)].

In addition, standard datasets such as [furg-fire-dataset](https://github.com/steffensbola/furg-fire-dataset) were also used for training and evaluation (and are included as a subset within the above datasets for [[Dunnings, 2018 - original data](https://collections.durham.ac.uk/downloads/r2d217qp536)]).

\* DOI for datsets - [http://doi.org/10.15128/r2d217qp536](http://doi.org/10.15128/r2d217qp536) and [http://doi.org/10.15128/r10r967374q](http://doi.org/10.15128/r10r967374q).

A download script ```download-dataset.sh``` is also provided which will create an additional ```dataset``` directory containing the training dataset (10.5Gb in size, works on Linux/MacOS).

![](https://github.com/tobybreckon/fire-detection-cnn/blob/master/images/slic-stages.png)

Original frame (left), Frame after superpixel segmentation (middle), Frame after superpixel fire prediction (right)

---

## Instructions to test pre-trained models:

To download and test the supplied code and pre-trained models (with \*\*TensorFlow 1.x / TFLearn 0.3.2 / OpenCV 4.x\*\* installed) do:

```

$ git clone https://github.com/tobybreckon/fire-detection-cnn.git

$ cd fire-detection-cnn

$ sh ./download-models.sh

$ python firenet.py models/test.mp4

$ python inceptionVxOnFire.py -m 1 models/test.mp4

$ python superpixel-inceptionVxOnFire.py -m 1 models/test.mp4

```

where ```-m x``` specifies the use of either of the \_InceptionV1OnFire, InceptionV3OnFire, InceptionV4OnFire\_

models for ```x``` in ```[1,3,4]```. By default it uses \_InceptionV1OnFire\_ if ```-m``` is not specified.

If by default you have \*\*TensorFlow 2.x\*\* installed on your system, then a workflow to make this repo work on your system is via a Tensorflow 1.x virtual environment (as TFLearn is not supported in TensorFlow 2.0 - this [issue](https://github.com/tflearn/tflearn/issues/1121)) is as follows:

```

$ virtualenv -p python3 ~/venv/tf-1.1.5-gpu

$ source ~/venv/tf-1.1.5-gpu/bin/activate

$ pip install tensorflow-gpu==1.15

$ pip install tflearn

$ pip install opencv-contrib-python

....

$ python3 firenet.py models/test.mp4

```

---

## Instructions to use pre-trained models with other frameworks:

To convert the supplied pre-trained models from TFLearn checkpoint format to protocol buffer (.pb) format (used by [OpenCV](http://www.opencv.org) DNN, [TensorFlow](https://www.tensorflow.org/), ...) and also tflite (used with [TensorFlow](https://www.tensorflow.org/)) do:

```

$ cd converter

$ python firenet-conversion.py

$ python inceptionVxOnFire-conversion.py -m 1

```

This creates a set of six ```.pb``` and ```.tflite``` files inside the ```converter``` directory (```firenet.xxx``` / ```inceptionv1onfire.xxx```/```sp-inceptionv1onfire.xxx``` for ```xxx``` in ```[pb, tflite]```). The ```inceptionVxOnFire-conversion.py``` can be similarly run with ```-m 3``` and ```-m 4``` to generate the same conversions for the \_InceptionV3OnFire\_ and \_InceptionV4OnFire\_ models respectively.

These alternative format files can then be validated with the [OpenCV](http://www.opencv.org) DNN module (OpenCV > 4.1.0-pre) and [TensorFlow](https://www.tensorflow.org/) against the original (tflearn) version from within the same directory, in order to check that they all produce the same output (up to 3 decimal places) as follows:

```

$ python firenet-validation.py

Load tflearn model from: ../models/FireNet ...OK

Load protocolbuf (pb) model from: firenet.pb ...OK

Load tflite model from: firenet.tflite ...OK

Load test video from ../models/test.mp4 ...

frame: 0 : TFLearn (original): [[9.999914e-01 8.576833e-06]] : Tensorflow .pb (via opencv): [[9.999914e-01 8.576866e-06]] : TFLite (via tensorflow): [[9.999914e-01 8.576899e-06]]: all equal test - PASS

frame: 1 : TFLearn (original): [[9.999924e-01 7.609045e-06]] : Tensorflow .pb (via opencv): [[9.999924e-01 7.608987e-06]] : TFLite (via tensorflow): [[9.999924e-01 7.608980e-06]]: all equal test - PASS

frame: 2 : TFLearn (original): [[9.999967e-01 3.373572e-06]] : Tensorflow .pb (via opencv): [[9.999967e-01 3.373559e-06]] : TFLite (via tensorflow): [[9.999967e-01 3.373456e-06]]: all equal test - PASS

frame: 3 : TFLearn (original): [[9.999968e-01 3.165212e-06]] : Tensorflow .pb (via opencv): [[9.999968e-01 3.165221e-06]] : TFLite (via tensorflow): [[9.999968e-01 3.165176e-06]]: all equal test - PASS

...

```

This can be similarly repeated with the ```inceptionVxOnFire-validation.py``` scripts with the options ```-m x``` for ```x``` in ```[1,3,4]``` for each of the InceptionVxOnFire models and similarly with the additional option ```-sp``` for each of the superpixel InceptionVxOnFire models (e.g. ```inceptionVxOnFire-validation.py -m 3 -sp``` validates the \_InceptionV3OnFire\_ superpixel model and so on). N.B. here the superpixel inceptionVxOnFire models are being validated against the whole image frame rather than superpixels just for simply showing consistent output between the original and converted models. Some ```FAIL```cases will be reported against this strict 3 decimal place criteria, but inspection often reveals a mildly larger ~0.1 difference (with the exception of the \_...V3-OnFire\_ and \_...V4-OnFire\_ caveat discussed above).

\*\*To convert to to other frameworks\*\* (such as PyTorch, MXNet, Keras, ...) from these tensorflow formats: - please see the extensive deep neural network model conversion tools offered by the [MMdnn](https://github.com/Microsoft/MMdnn) project.

---

## Example video:

[![Examples](https://github.com/tobybreckon/fire-detection-cnn/blob/master/images/slic-ex.png)](https://youtu.be/RcNj8aMDer4)

Video Example - click image above to play.

---

## References:

If you are making use of this work in any way (including our pre-trained models or datasets), \_you must please\_ reference the following articles in any report, publication, presentation, software release

or any other associated materials:

[Experimentally defined Convolutional Neural Network Architecture Variants for Non-temporal Real-time Fire Detection](https://breckon.org/toby/publications/papers/dunnings18fire.pdf)

(Dunnings, Breckon), In Proc. International Conference on Image Processing, IEEE, 2018.

```

@InProceedings{dunnings18fire,

author = {Dunnings, A. and Breckon, T.P.},

title = {Experimentally defined Convolutional Nerual Network Architecture Variants for Non-temporal Real-time Fire Detection},

booktitle = {Proc. International Conference on Image Processing},

pages = {1558-1562},

year = {2018},

month = {September},

publisher = {IEEE},

doi = {10.1109/ICIP.2018.8451657},

keywords = {simplified CNN, deep learning, fire detection, real-time, non-temporal, non-stationary visual fire detection, FireNet, InceptionV1OnFire},

}

```

[Experimental Exploration of Compact Convolutional Neural Network Architectures for Non-temporal Real-time Fire Detection](https://breckon.org/toby/publications/papers/samarth19fire.pdf)

(Samarth, Bhowmik, Breckon), In Proc. International Conference on Machine Learning Applications, IEEE, 2019.

```

@InProceedings{samarth19fire,

author = {Samarth, G. and Bhowmik, N. and Breckon, T.P.},

title = {Experimental Exploration of Compact Convolutional Neural Network Architectures for Non-temporal Real-time Fire Detection},

booktitle = {Proc. International Conference on Machine Learning Applications},

pages = {653-658},

year = {2019},

month = {December},

publisher = {IEEE},

doi = {10.1109/ICMLA.2019.00119},

keywords = {fire detection, CNN, deep-learning real-time, non-temporal, InceptionV3OnFire, InceptionV4OnFire},

}

```

In addition the (very permissive) terms of the [LICENSE](LICENSE) must be adhered to.

### Acknowledgements:

Atharva (Art) Deshmukh (Durham University, \_github and data set collation for publication\_ for [Dunnings/Breckon, 2018] work).

---