ImageSig: A signature transform for ultra-lightweight image recognition (Supplementary)

Mohamed R. Ibrahim[1]
[1]The Alan Turing Institute
London, UK
mibrahim@turing.ac.uk

Terry Lyons[1,2]
[2] Mathematical Institute, Oxford University
Oxford, UK
terry.lyons@maths.ox.ac.uk

1. ImageSig as a python module

Here is an example for training and inference using ImageSig as a python module.

```
### Training

from imagesig import image_signatrue
SIG_DEPTH = 4
IMAGE_SIZE = (64,64)
TRAIN_DIR = "data/training_set"
TEST_DIR = "data/test_set

train_sig = image_signature (
            image_dir = TRAIN_DIR,
            depth = SIG_DEPTH,
            image_size = IMAGE_SIZE,
            flatten=FLATTEN,
            log_sig = False,
            two_direction = True,
            augment_flip_horizontal = True,
            augment_flip_vertical =True,
            augment_brightness= True
            )
test_sig = image_signature (
            image_dir = TEST_DIR,
            depth = SIG_DEPTH,
            image_size = IMAGE_SIZE,
            flatten=FLATTEN,
            log_sig = False,
            two_direction=True
            )
_,x_train,y_train = train_sig.read_dir()
_,x_test,y_test = test_sig.read_dir()

......  Model architecture ......

### Inference

from imagesig import imagesig_predict
model = "checkpoint"
image = "fire.jpg"
pred = imagesig_predict(image,model)
```

2. Dataset details and accessibility

Table 1 shows how the image samples are distributed per each class in training and testing for all datasets. Test set is the data subset where all models are evaluated, in which the model never seen during training and validation of each epoch.

Fire dataset access: To be made available upon request.

Fog dataset access: Can be requested from WeatherNet's authors: https://www.mdpi.com/2220-9964/8/12/549

Concrete crack dataset access: https://data.mendeley.com/datasets/5y9wdsg2zt/2

CelebA dataset access: https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

3. Implementation details

All codes are provided for our implementations in the ZIP file. Here is a brief description of our implementations:

Image signature: We have created our own python class for implementing and augmenting signature to images as paths of streams, as described in this paper, called image signature. We built image signature based on computing signature from two different python packages as backends for image signature so-called Signatory [1] and iisignature [2]. We did not observe a noticeable effect on which backend we used on the performance of a given model. Accordingly, We relied mainly on Signatory for all models, however, for edge devices such as RaspberryPi we used iisignature, given that signatory depends on Pytorch for acceleration whereas iisignature outputs numpy arrays that can

Dataset	Subset	Classes	Sample	(α)
Fire	Training	Fire	4,097	1.84
		No fire	11,055	0.68
	Test	Fire	960	-
		No fire	2,800	-
Fog	Training	Fog	589	2.93
		No fog	2,860	0.60
	Test	Fog	129	-
		No fog	769	-
Concrete crack	Training	Negative	16,000	1.00
		Positive	16,000	1.00
	Test	Negative	4,000	-
		Positive	4,000	-
CelebA	Training	Multi-task	162,770	-
	Validation	Multi-task	19,867	-
	Test	Multi-task	19,962	-

Table 1. Data sample distribution and class weight.

fit both tensorflow and Pytorch models without the need for converting torch tensors to TF tensors. The main models are computed with input resolution of 64 X 64 and signature depth = 4. However, in the ablation studies, we show the effect of various signature depth and image input size on the performance of the model.

Data augmentations: All models are trained with the same data augmentation techniques such as normalizations, horizontal flipping, and changing colour brightness.

Fig. 1 shows an example for the augmented images and their respective signatures for training.

ImageSig FC-based Mode: We trained several models with different hyper-parameters that only rely on a single FC layer. This FC layer comprises 50 neurons and is activated by a ReLU function. Models are trained with Adam optimiser of batch size 3000 and for 300 epochs. In ablation studies, we show the effect of the selected different hyperparameters on the performance of the model such as changing the number of neurons or batch size.

ImageSig convolution-based model: After computing signature, we used a simple convolution block comprised of two CNN 1D layers of feature maps of 32 and 64 respectively and a kernel size of 3, in which both layers are activated by a ReLU function. Each layer is followed by a Max-pooling layer of kernel size 3. After the last pooling layer, the model is flattened and feedforward to a single FC layer of 50 neurons that is activated by a ReLU function before the final softmax layer. The model is trained based on Adam optimisation [3] and a batch size of 3000 for 300 epochs.

ImageSig Attention-based model: after the signature input, we added a convolution block implemented similar to the aforementioned architecture. After the convolution block, we normalized the final output of the last pooling layer and passed it to a multi-head attention layer of eight heads. Afterwards, we add a skip connection that connect the attention outputs and the pooling layer. For each transformer layer (50 in total), we added a normalization and a MLP layers similar to transformer units [64,64]. Finally, we added an FC layer of 50 neurons. The model is trained with 64 projection dimensions that is equivalent to dimension of a given input. All layers are activated by a ReLU function, except the softmax output layer. The model is trained with AdamW optimiser with a learning rate of 0.001, and a decay of 0.0001 with a batch-size of 300 and 3000 epochs.

Convolution base models: All image convolution models are implemented via transfer learning on the given custom dataset with ImageNet weights. After truncated the models, we trained an FC and output layers with the same hyperparameters of the aforementioned architectures. All models have converged when trained for 20 epochs and patch size 1024.

ViT base models: We trained transformer models from scratch on the mentioned datasets. We followed the implementation details for ViT model in their original paper closely. The model is optimised based on AdamW and patch size of 1024 and trained for 300 epochs.

4. Further Results

From Fig. 2-7, we show a sample of predictions of our models on the different datasets showing true and false prediction.

5. Further ablation studies

The effect of the number of neurons:

We noticed a direct effect of the number of neurons of the fully connected layer on the performance of a given model and its number of parameters. Table 2 summarises the results of 10 models to investigate the effect of the number of neurons. All models are trained with a single fully connected layer of different numbers of neurons. For every 5 models, the image resolution and the depth of the signature are kept constant, whereas the number of neurons has been changed between 25-200 neurons. The table shows that, at a given image resolution and signature depth, by doubling the number of neurons, a slight increase $(< 1\%)$ in the accuracy can be achieved, bearing in mind the double increase of the mode's parameters and consequently its disk size. We found that optimal results as a trade-off between the accuracy and model size can be achieved within the choice of 50 neurons as a hyperparameter of the fully connected layer.

The effect of batch size:

In table 3, we show the effect of batch size on accuracy. From our experiments, larger batch size allows stability in

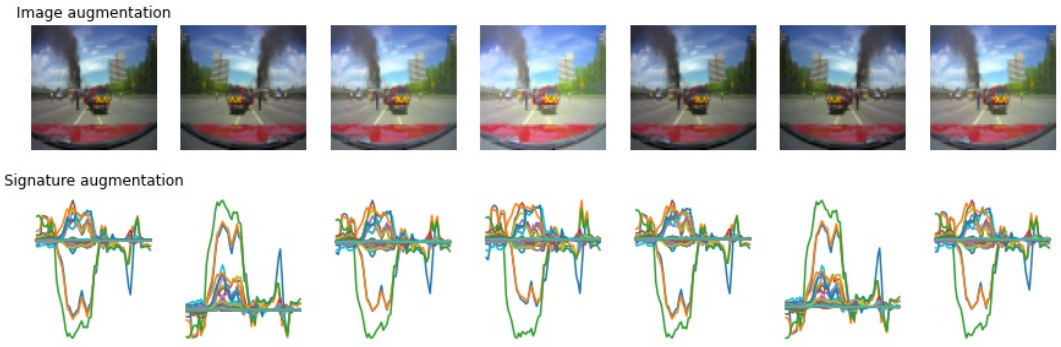

Figure 1. Sample of signature augmentations.

Resolution	Neurons	acc (%)	Params	Model size	Training time
(32,32), Depth = 4	25	90.53	96,072	1,180 KB	2.34 min
	50	91.54	192,152	2,278 KB	4.16 min
	100	92.23	384,302	4,638 KB	2.38 min *
	150	92.10	576,452	6,945 KB	2.35 min *
	200	93.00	768,602	9,253 KB	2.35 min *
(64,64), Depth = 4	25	91.19	192,077	2,332 KB	3.99 min*
	50	92.36	384,152	4,528 KB	5.22 min
	100	92.39	768,302	9,246 KB	4.02 min*
	150	91.99	1,151,452	13,857 KB	4.02 min*
	200	92.87	1,536,602	18,469 KB	4.04 min*

Table 2. Comparing the effects of the number of neurons of the single FC layer on the performance.All models are trained with a fixed signature depth (N=4).

Figure 2. Predicted values: fire, no fog.

Figure 3. Predicted values: No fire, fog.

training and convergence. Giving that the model requires very small memory, training it with a large batch size can be achieved with a minimal shared memory in comparison to other models such as ResNet or MobieNetV2.

Furthermore, Table 4 shows the effect of the batch size on the overall performance of a given model in relation to the signature depth. The table shows a better results at a batch size of 3000 when compared to a batch size of 1024, for the same image resolution and signature depth.

Figure 4. A sample of prediction on the fire dataset. Model predictions (green: correct, red: incorrect).

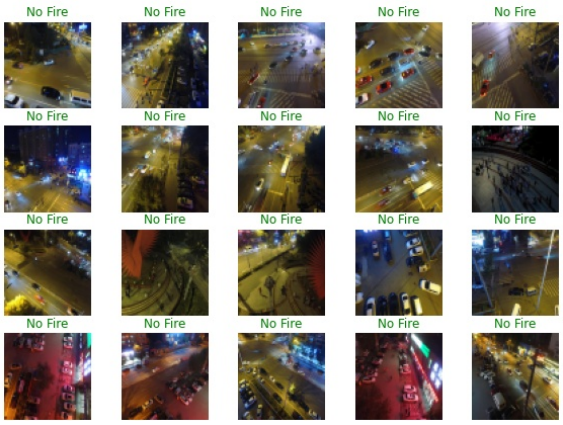

Figure 5. A sample of prediction on the fire dataset. Model predictions (green: correct, red: incorrect).

Batch size	64	128	256	512	1024
acc (%)	79.01	88.13	90.36	92.52	94.38

Table 3. The effect of batch size on acc on fire dataset

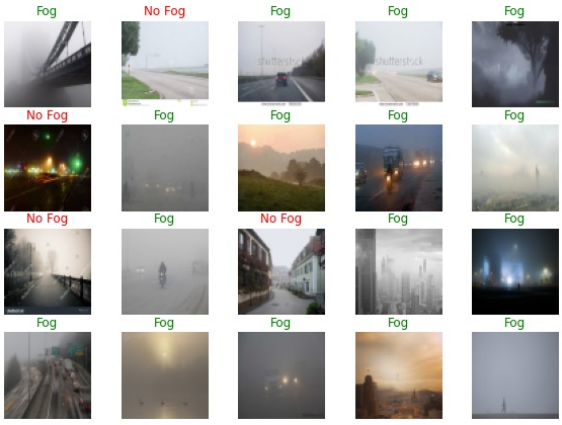

Figure 6. A sample of prediction on the fog dataset. Model predictions (green: correct, red: incorrect).

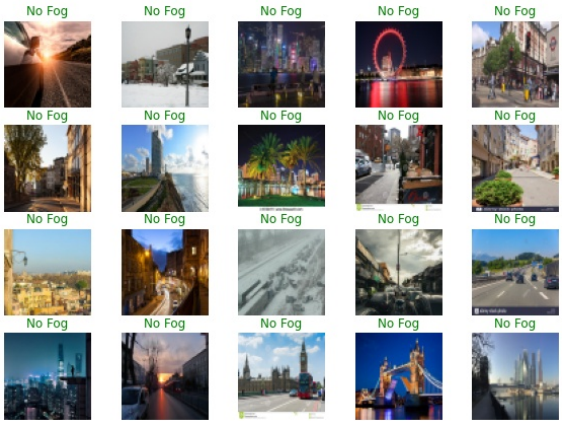

Figure 7. A sample of prediction on the fog dataset. Model predictions (green: correct, red: incorrect).

References

[1] Patrick Kidger and Terry Lyons. Signatory: differentiable computations of the signature and logsignature transforms, on both CPU and GPU. In *International Conference on Learning Representations*, 2021. https://github.com/patrick-kidger/signatory. 1

[2] Jeremy Reizenstein and Benjamin Graham. Algorithm 1004: The iisignature library: Efficient calculation of iterated-integral signatures and log signatures. *ACM Transactions on Mathematical Software (TOMS)*, 2020. 1

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015. 2

Batch size	Sig. Depth	acc (%)	Training time (%)
1024	1	83.45	1.54 min
	2	89.17	1.94 min
	3	90.80	3.21 min
	4	**92.36**	5.22 min
	5	92.18	17.42 min
3000	1	80.29	0.72 min
	2	89.38	1.02 min
	3	91.62	1.88 min
	4	93.40	4.47 min
	5	**93.78**	12.23 min

Table 4. Comparing the effects of different batch size on the performance. All models are trained with a single FC layer of 50 neurons for 200 epochs on fire recognition dataset.