Homework 05 – Design

Arthur J. Redfern
arthur.redfern@utdallas.edu

# 1 Reading

1. Design
Motivation: understand xNN design – ***Done***

2. Understanding LSTM networks
Motivation: an alternative presentation of RNNs and variants – ***Done***

3. Attention and augmented recurrent neural networks
Motivation: an alternative presentation of attention – ***Done***

4. The illustrated transformer
Motivation: an alternative presentation of self-attention – ***Done***

5. The annotated transformer
Motivation: a code walkthrough of self-attention – ***Done***

6. [Optional] ResNet / ResNeXt
Deep residual learning for image recognition – ***Done***
Identity mappings in deep residual networks – ***Done***
Aggregated residual transformations for deep neural networks – ***Done***

7. [Optional] Neural architecture search – ***Done***

# 2 Theory

8. The receptive field size at the input to the global average pooling layer for ResNet 50. – ***receptive_field_size/rfs_resenet50.py, out_resenet50.txt***

```
ResNet without bottleneck
Each block: 3x3 -> 3x3

receptive_field_size= 1

Filter
receptive_field_size= 1 + 3 - 1 = 3
receptive_field_size= 3 + 3 - 1 = 5
receptive_field_size= 5 + 3 - 1 = 7
receptive_field_size= 7 + 3 - 1 = 9
receptive_field_size= 9 + 3 - 1 = 11
receptive_field_size= 11 + 3 - 1 = 13
Pool
receptive_field_size= 13 x 2 - 2 - 1 = 25

Filter
receptive_field_size= 25 + 3 - 1 = 27
receptive_field_size= 27 + 3 - 1 = 29
receptive_field_size= 29 + 3 - 1 = 31
receptive_field_size= 31 + 3 - 1 = 33
receptive_field_size= 33 + 3 - 1 = 35
receptive_field_size= 35 + 3 - 1 = 37
receptive_field_size= 37 + 3 - 1 = 39
receptive_field_size= 39 + 3 - 1 = 41
receptive_field_size= 41 + 3 - 1 = 43
receptive_field_size= 43 + 3 - 1 = 45
receptive_field_size= 45 + 3 - 1 = 47
receptive_field_size= 47 + 3 - 1 = 49
Pool
receptive_field_size= 49 x 2 - 2 - 1 = 97

Filter
receptive_field_size= 97 + 3 - 1 = 99
receptive_field_size= 99 + 3 - 1 = 101
receptive_field_size= 101 + 3 - 1 = 103
receptive_field_size= 103 + 3 - 1 = 105
receptive_field_size= 105 + 3 - 1 = 107
receptive_field_size= 107 + 3 - 1 = 109
receptive_field_size= 109 + 3 - 1 = 111
receptive_field_size= 111 + 3 - 1 = 113
Pool
receptive_field_size= 113 x 2 - 2 - 1 = 225

Filter
receptive_field_size= 225 + 3 - 1 = 227
receptive_field_size= 227 + 3 - 1 = 229
receptive_field_size= 229 + 3 - 1 = 231
receptive_field_size= 231 + 3 - 1 = 233
receptive_field_size= 233 + 3 - 1 = 235
receptive_field_size= 235 + 3 - 1 = 237
```

```
TAIL

Pool
receptive_field_size= 237 x 2 - 2 - 1 = 473
receptive_field_size= 473 x 2 - 2 - 1 = 945
Filter
receptive_field_size= 945 + 7 - 1 = 951

receptive_field_size= 951


ResNet with bottleneck
Each block: 1x1 -> 3x3 -> 1x1

receptive_field_size= 1

Filter
receptive_field_size= 1 + 3 - 1 = 3
receptive_field_size= 3 + 3 - 1 = 5
receptive_field_size= 5 + 3 - 1 = 7
Pool
receptive_field_size= 7 x 2 - 2 - 1 = 13

Filter
receptive_field_size= 13 + 3 - 1 = 15
receptive_field_size= 15 + 3 - 1 = 17
receptive_field_size= 17 + 3 - 1 = 19
receptive_field_size= 19 + 3 - 1 = 21
receptive_field_size= 21 + 3 - 1 = 23
receptive_field_size= 23 + 3 - 1 = 25
Pool
receptive_field_size= 25 x 2 - 2 - 1 = 49

Filter
receptive_field_size= 49 + 3 - 1 = 51
receptive_field_size= 51 + 3 - 1 = 53
receptive_field_size= 53 + 3 - 1 = 55
receptive_field_size= 55 + 3 - 1 = 57
Pool
receptive_field_size= 57 x 2 - 2 - 1 = 113

Filter
receptive_field_size= 113 + 3 - 1 = 115
receptive_field_size= 115 + 3 - 1 = 117
receptive_field_size= 117 + 3 - 1 = 119

TAIL

Pool
receptive_field_size= 119 x 2 - 2 - 1 = 237
receptive_field_size= 237 x 2 - 2 - 1 = 473
Filter
receptive_field_size= 473 + 7 - 1 = 479

receptive_field_size= 479
```

# 3 Practice

9. Understand all lines of code in the following example (https://github.com/arthurredfern/UTDallas-CS-6301-CNNs/blob/master/Code/xNNs_Code_031_CIFAR_ResNetV2b.py) – ***Done***

10. Practice with MobileNet V2 – ***receptive_field_size/rfs_mobilenetv2.py, out_mobilenetv2.txt***

- Receptive field size at GAP

```
MobileNet V2 with bottle neck
Each block: 1x1 -> 3x3 -> 1x1

receptive_field_size= 1

Level2

Filter
receptive_field_size= 1 + 3 - 1 = 3
receptive_field_size= 3 + 3 - 1 = 5
receptive_field_size= 5 + 3 - 1 = 7
receptive_field_size= 7 + 3 - 1 = 9
Pool
receptive_field_size= 9 x 2 - 2 - 1 = 17

Level1

Filter
receptive_field_size= 17 + 3 - 1 = 19
receptive_field_size= 19 + 3 - 1 = 21
receptive_field_size= 21 + 3 - 1 = 23
receptive_field_size= 23 + 3 - 1 = 25
receptive_field_size= 25 + 3 - 1 = 27
receptive_field_size= 27 + 3 - 1 = 29
receptive_field_size= 29 + 3 - 1 = 31
Pool
receptive_field_size= 31 x 2 - 2 - 1 = 61

Level0

Filter
receptive_field_size= 61 + 3 - 1 = 63
receptive_field_size= 63 + 3 - 1 = 65
receptive_field_size= 65 + 3 - 1 = 67
receptive_field_size= 67 + 3 - 1 = 69
receptive_field_size= 69 + 3 - 1 = 71

TAIL
Filter
receptive_field_size= 71 + 3 - 1 = 73

receptive_field_size= 73
```

MobileNet V2 has almost identical structure as ResNet with bottlenecks.

- Feature map size and memory for each linear layer

Parameters for inverted residual blocks:

- expand_dim
- squeeze_dim
- strides

Input = 28x28x3 tensor

***Tail***

Conv 3x3, 16 filters => 28 x 28 x 16 : 12544 x 32 bits

***Level 1***

Inverted Residual x 4 :
Conv 1x1 => 28 x 28 x 64 : 50176 x 32 bits
DConv 3x3 => 28 x 28 x 64 : 50176 x 32 bits
Conv 1x1 => 28 x 28 x 16 : 12544 x 32 bits

***Level 2***

Inverted Residual x 6 :
=> 14 x 14 x 128 : 25088 x 32 bits
=> 14 x 14 x 128 : 25088 x 32 bits
=> 14 x 14 x 64 : 12544 x 32 bits

***Level 3***

Inverted Residual x 3 :
=> 7 x 7 x 256 : 12544 x 32 bits
=> 7 x 7 x 256 : 12544 x 32 bits
=> 7 x 7 x 128 : 6272 x 32 bits

Conv 1x1 => 7 x 7 x 256 : 1254432 bits

***Head***

GAP => 1 x 256 : 256 x 32 bits

Dense => 1 x 200

The maximum feature map size is in the first 1x1 expansion layer in the first block:
28 x 28 x 64 : 50176 x 32 bits.

- Filter coefficient size and memory for a complete block at each level

  Input = 28x28x3 tensor

  ***Tail***

| | |
|---|---|
| Conv 3x3, 16 filters | => 3 x 3 x 3 x 16 : 12544 x 32 bits |

  ***Level 1***

| | |
|---|---|
| Inverted Residual x 4 : | |
| Conv 1x1 | => 1 x 1 x 16 x 64 : 1024 x 32 bits |
| DConv 3x3 | => 3 x 3 x 64 : 576 x 32bits |
| Conv 1x1 | => 1 x 1 x 64 x 16 : 1024 x 32bits |

  ***Level 2***

| | |
|---|---|
| Inverted Residual x 6 : | |
| | => 1 x 1 x 64 x 128 : 8192 x 32 bits |
| | => 3 x 3 x 128 : 1152 x 32 bits |
| | => 1 x 1 x 128 x 64 : 8192 x 32 bits |

  ***Level 3***

| | |
|---|---|
| Inverted Residual x 3 : | |
| | => 1 x 1 x 128 x 256 : 32768 x 32 bits |
| | => 3 x 3 x 256 : 2304 x 32 bits |
| | => 1 x 1 x 256 x 128 : 32768 x 32 bits |
| Conv 1x1 | => 1 x 1 x 128 x 256 : 32768 x 32 bits |

  ***Head***

| | |
|---|---|
| GAP | => 0 : 0 x 32 bits |
| Dense | => 256 x 200 : 51200 x 32 bits |

  The model parameters become denser as the network progresses.
  The final level has the most filter memory requirements, 32768 x 32 bits.

- MACs for a complete block at each level

  Input = 28x28x3 tensor

  ***Tail***

  Conv 3x3, 16 filters

  ***Level 1***

  Inverted Residual x 4 :
  Conv 1x1 => 1 x 1 x 16 x 64 x 28 x 28 : 802K MACs
  DConv 3x3 => 3 x 3 x 64 x (28+2) x (28+2) : 518K MACs
  Conv 1x1 => 1 x 1 x 64 x 16 x 28 x 28 : 802K MACs

  ***Level 2***

  Inverted Residual x 6 :
  => 1 x 1 x 64 x 128 x 14 x 14 : 1.6M MACs
  => 3 x 3 x 128 x (14+2) x (14+2) : 294K MACs
  => 1 x 1 x 128 x 64 x 14 x 14 : 1.6M MACs

  ***Level 3***

  Inverted Residual x 3 :
  => 1 x 1 x 128 x 256 x 7 x 7 : 1.6M MACs
  => 3 x 3 x 256 (7+2) x (7+2) : 186K MACs
  => 1 x 1 x 256 x 128 x 7 x 7 : 1.6M MACs

  Conv 1x1

  ***Head***

  GAP => 0

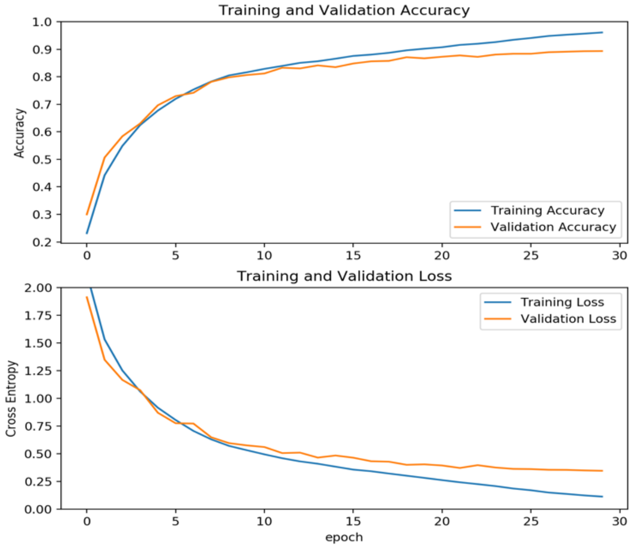
  Dense => 256 x 200

  The MACs increase as the network progresses, but plateau as the dimension of the feature maps reduce. The second layer has the most MACs and the first has the fewest.

- From the perspective of increasing receptive field size, minimizing filter memory and minimizing MACs, level 3 or 4 is the best to repeat blocks within.

11. Model Implementation – ***model_implementation/main.py***

```
Training model mobilenet_v2...
Epoch: 0 lr: 1e-05
Epoch: 1 lr: 0.00020800000000000001
Epoch: 2 lr: 0.00040600000000000006
Epoch: 3 lr: 0.000604
Epoch: 4 lr: 0.0008020000000000001
Epoch: 5 lr: 0.001
Epoch: 6 lr: 0.0009978803340062175
Epoch: 7 lr: 0.0009915304127600722
Epoch: 8 lr: 0.000980977427599198
Epoch: 9 lr: 0.0009662665680261777
Epoch: 10 lr: 0.0009474608282001546
Epoch: 11 lr: 0.0009246407371861739
Epoch: 12 lr: 0.0008979040141173614
Epoch: 13 lr: 0.0008673651497465943
Epoch: 14 lr: 0.0008331549161795198
Epoch: 15 lr: 0.0007954198068883229
Epoch: 16 lr: 0.0007543214094041876
Epoch: 17 lr: 0.0007100357133746822
Epoch: 18 lr: 0.0006627523569490682
Epoch: 19 lr: 0.0006126738147186335
Epoch: 20 lr: 0.0005600145306894063
Epoch: 21 lr: 0.0005050000000000001
Epoch: 22 lr: 0.0004478658033168113
Epoch: 23 lr: 0.00038885659804143894
Epoch: 24 lr: 0.0003282250706501301
Epoch: 25 lr: 0.0002662308546514956
Epoch: 26 lr: 0.00020313941879596703
Epoch: 27 lr: 0.0001392209302978512
Epoch: 28 lr: 7.474909793784162e-05
Epoch: 29 lr: 1.0000000000000062e-05
Training complete.
```



```
Test loss:      0.3462624225435378
Test accuracy:  0.8933
```