COMP4107 Project Report

Christian Abbott TODO number

Basim Ramadhan 100 901 646

1 Running Our Program

TODO create a readme

```
make prepare-venv
source env/bin/activate
python train.py &
python evaluate.py &
tensorboard --logdir=$(pwd)/logs
```

The commands above will perform the following:

1. Create a virtual environment with the required libraries.
2. Enter the virtual environment.
3. Initialize a new classifier and start training it.
4. Repeatedly evaluate the classifier's accuracy so far.
5. Start TensorBoard to monitor the classifier's accuracy.

2 The Problem Being Solved

For our project we investigated and compared the performance of the state-of-the-art in CNN's for image recognition with the ImageNet dataset. We decided to use the Tiny ImageNet[1] dataset for our project because the entire ImageNet dataset proved to be too computationally expensive for us when we tried using it in the beginning. This reduced dataset contains 200 classes to recognize, as opposed to the 1000 classes in full ImageNet. We wanted to learn about different researchers efforts in tackling the ImageNet dataset. We wanted to create our own classifier for recognizing images from the Tiny ImageNet dataset based on what those researchers did. The classifier models we implemented and compared are "reduced" versions of the originals. In other words, we tried to compress them and retain classification accuracy.

2.1 Experimental Methodology

2.2 Baseline CNN Architecture

For our experiments, we implemented classifier models based on the works of different researchers. In the beginning, we implemented a very simple CNN on our own. This simple model will provide a baseline to compare other models to.

[1] https://tiny-imagenet.herokuapp.com/

2.3 SimpleNet

For our experiments, we implemented classifier models based on the works of different researchers. In the beginning, we implemented a very simple CNN on our own. This simple model will provide a baseline to compare other models to.

2.4 VGG 16

For our experiments, we implemented classifier models based on the works of different researchers. In the beginning, we implemented a very simple CNN on our own. This simple model will provide a baseline to compare other models to.

2.5 AlexNet

For our experiments, we implemented classifier models based on the works of different researchers. In the beginning, we implemented a very simple CNN on our own. This simple model will provide a baseline to compare other models to.

3 Our Best Classifier

Architecture:

- Convolution 3x3
- Max Pooling 2x2
- Fully connected 1024
- Fully connected 200

3.1 Hyperparameters

Learning Rate $0.00TODO$

4 Limitations

1. Limited GPU power.
2. Limited time means we couldn't work out all the issues.

5 Conclusion & Future Work

The end!

6 References

- https://arxiv.org/abs/1608.06037
(SimpleNet)
- https://arxiv.org/abs/1409.1556
(VGG 16)
- https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
(AlexNet)
- https://arxiv.org/abs/1412.6071
(Fractional Pooling)

- http://scott.fortmann-roe.com/docs/BiasVariance.html
- TODO
- TODO
- TODO

7 Figures

7.1 Baseline CNN Architecture & Results

Architecture:
TODO add RELU to all architecture boxes

- Convolution: 3x3
- Max Pooling: 2x2
- Fully connected: 1024
- Fully connected: 200
- Softmax

7.2 SimpleNet Architecture & Results

Architecture:
TODO I used simplenetC

- Convolution: 2x2
- Convolution: 2x2
- Convolution: 1x1
- Convolution: 2x2
- Max Pooling: 2x2
- Convolution: 2x2
- Convolution: 2x2
- Max Pooling: 2x2
- Convolution: 2x2
- Max Pooling: 2x2
- Convolution: 2x2
- Convolution: 2x2
- Max Pooling: 2x2
- Convolution: 2x2
- Convolution: 1x1
- Convolution: 1x1
- Max Pooling: 2x2
- Convolution: 2x2
- Max Pooling: 2x2
- Convolution: 1x1
- Softmax

7.3 VGG16 Architecture & Results

Architecture:

- Convolution: 2x2
- Convolution: 2x1
- Convolution: 1x2
- Max Pooling: 2x2
- Convolution: 2x2
- Convolution: 2x2
- Convolution: 2x2
- Max Pooling: 2x2
- Convolution: 2x2
- Convolution: 2x1
- Convolution: 1x2
- Max Pooling: 2x2
- Fully Connected: 200
- Softmax

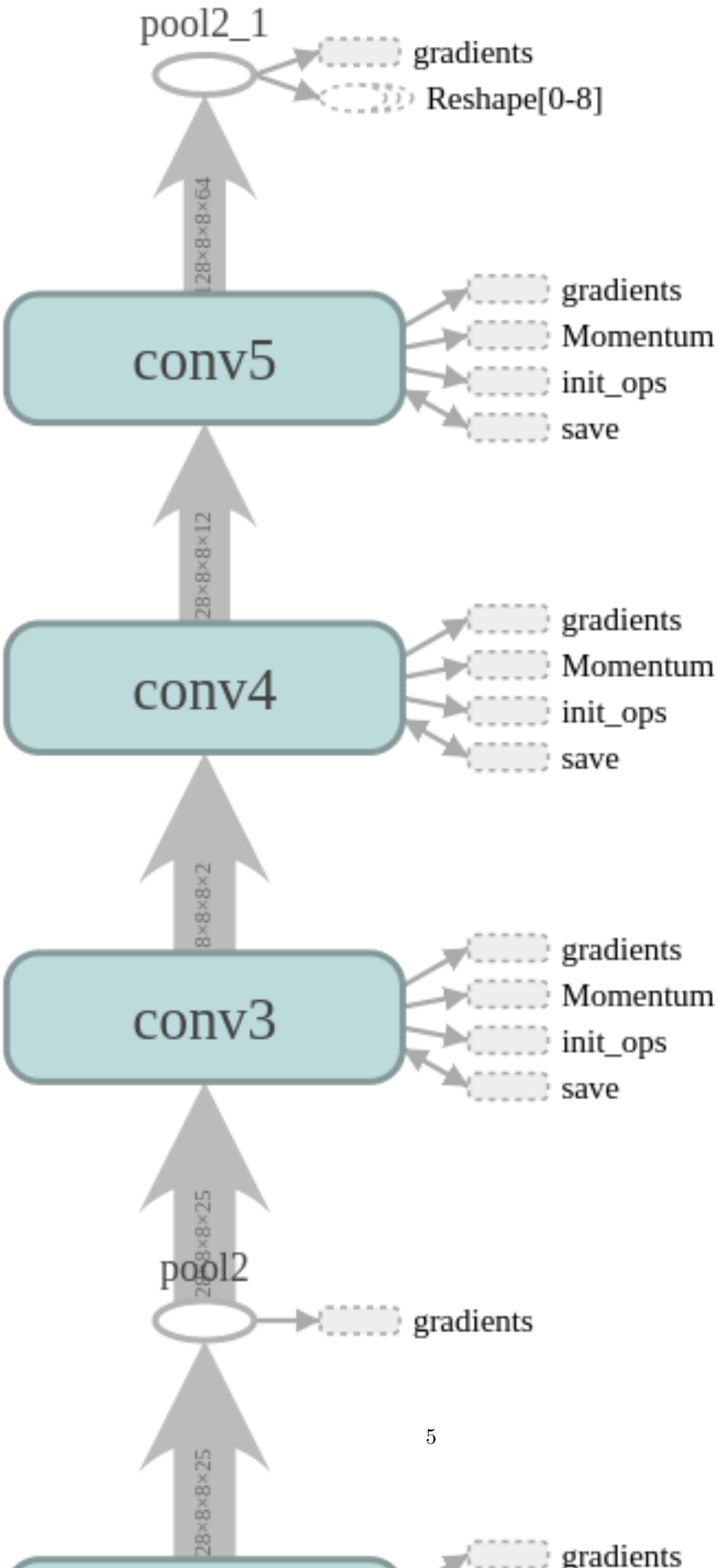
7.4 AlexNet Architecture & Results

TODO I'm using XL **Architecture:**

- Convolution: 11x11, stride 4
- Convolution: 5x5
- Max Pooling: 3x3, stride 2
- Convolution: 3x3
- Convolution: 3x3
- Convolution: 3x3
- Max Pooling: 3x3
- Convolution: 3x3
- Convolution: 3x3
- Convolution: 3x3
- Max Pooling: 3x3
- Fully Connected: 4096
- Fully Connected: 4096
- Fully Connected: 200
- Softmax

7.5 Computational Graph

We used TensorBoard to visualize our final model. Unfortunately, the graph wasn't as neatly organized as we would have liked. We hope the following snippets from the computational graph are useful:


pool2_1
gradients
Reshape[0-8]
128×8×8×64
conv5
gradients
Momentum
init_ops
save
28×8×8×12
conv4
gradients
Momentum
init_ops
save
8×8×8×2
conv3
gradients
Momentum
init_ops
save
28×8×8×25
pool2
gradients
28×8×8×25
gradients