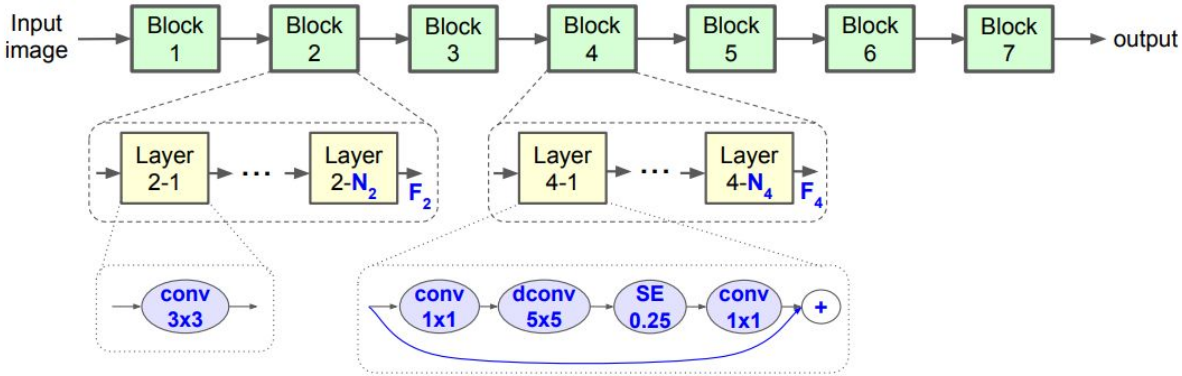
MNASNet:

Platform-Aware Neural Architecture Search for Mobile

Gilberto Barrientos
George Gomez
Robert Wallace

Quick Summary of Paper

- Proposal - automated neural architecture search approach for designing mobile CNN
- Step away from using FLOPS and instead to measure latency directly on real world mobile devices
- Factorized Hierarchical Search Space



Cost Function/Search Algorithm

- Reinforcement learning is used to find pareto optimal solution.
- Map each CNN in the search space as a list of tokens

		Inference Latency	Top-1 Acc.
w/o SE	MobileNetV2	75ms	72.0%
	NASNet	183ms	74.0%
	MnasNet-B1	77ms	74.5%
w/ SE	MnasNet-A1	78ms	75.2%
	MnasNet-A2	84ms	75.6%

$$J = E_{P(a_{1:T};\theta)}[R(m)]$$

$$ACC(m)$$

$$LAT(m)$$

Project Overview

Analysis of MNASnet_B1 network with CIFAR10 and FashionMNIST

- Implement MNASnet model via Pytorch
- Utilize new dataset:
  - FashionMNIST
    - Grayscale images of clothing
    - Single channel
  - CIFAR10
    - 10 classes of images such as [airplane,automobile, bird,cat,deer,dog,frog,horse,ship,truck]
    - Three channel
- Modify existing model
- Train and Evaluate - Google Colab
  - Accuracy
  - Latency

Model Modifications

- CIFAR10 Input matches that of ImageNet (3 channels)
  - Adjust classification layer
- FashionMNIST requires modification.
  - Adjust input layer and classification layer

```
MNASNet(
  (layers): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
(classifier): Sequential(
  (0): Dropout(p=0.2, inplace=True)
  (1): Linear(in_features=1280, out_features=1000, bias=True)
```

Before (ImageNet)

```
(model): MNASNet(
  (layers): Sequential(
    (0): Conv2d(1, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
(classifier): Sequential(
  (0): Dropout(p=0.2, inplace=True)
  (1): Linear(in_features=1280, out_features=10, bias=True)
)
```

After (FashionMNIST)

```
self.model.layers[0] = nn.Conv2d(in_channels, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
self.model.classifier[1] = nn.Linear(1280, 10)
```

Modification Code

Training

- Epochs: 5
- Training Set Size: 10000
- SGD optimizer
- Cross Entropy Loss Criterion
- Learning Rate: 0.01
- Batch Size: 32

```
Train Epoch: 4 [0/10000 (0%)]   Loss: 0.522517
Train Epoch: 4 [512/10000 (5%)] Loss: 0.468161
Train Epoch: 4 [1024/10000 (10%)]       Loss: 0.514315
Train Epoch: 4 [1536/10000 (15%)]       Loss: 0.664150
Train Epoch: 4 [2048/10000 (20%)]       Loss: 0.287251
Train Epoch: 4 [2560/10000 (26%)]       Loss: 0.498152
Train Epoch: 4 [3072/10000 (31%)]       Loss: 0.537857
Train Epoch: 4 [3584/10000 (36%)]       Loss: 0.547040
Train Epoch: 4 [4096/10000 (41%)]       Loss: 0.679059
Train Epoch: 4 [4608/10000 (46%)]       Loss: 0.467151
Train Epoch: 4 [5120/10000 (51%)]       Loss: 0.489341
Train Epoch: 4 [5632/10000 (56%)]       Loss: 0.768975
Train Epoch: 4 [6144/10000 (61%)]       Loss: 0.873401
Train Epoch: 4 [6656/10000 (66%)]       Loss: 0.674739
Train Epoch: 4 [7168/10000 (72%)]       Loss: 0.415817
Train Epoch: 4 [7680/10000 (77%)]       Loss: 0.435732
Train Epoch: 4 [8192/10000 (82%)]       Loss: 0.434942
Train Epoch: 4 [8704/10000 (87%)]       Loss: 0.573434
Train Epoch: 4 [9216/10000 (92%)]       Loss: 0.389902
Train Epoch: 4 [9728/10000 (97%)]       Loss: 0.592647
Train set: Average loss: 0.0171, Accuracy 7924/10000 (79.24%)
```

Training for FashionMNIST from pretrained model

Evaluation

- Evaluation Set Size: 10000
- Benchmark 10 Inferences
  - Average for latency

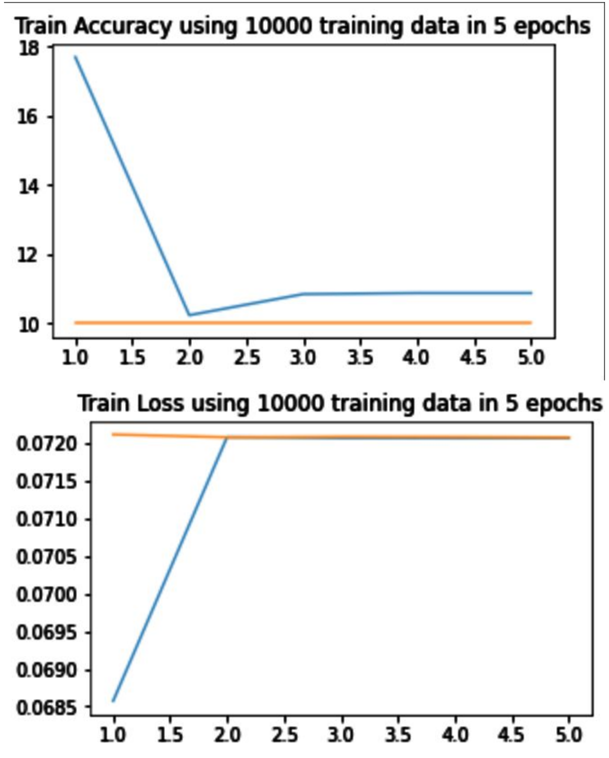
```
timer = benchmark.Timer(stmt="run_inference(model, inputs)",
                        setup="from __main__ import run_inference",
                        globals={
                            "model": model,
                            "inputs": inputs
                        })
```

Code Snippet of Benchmarking

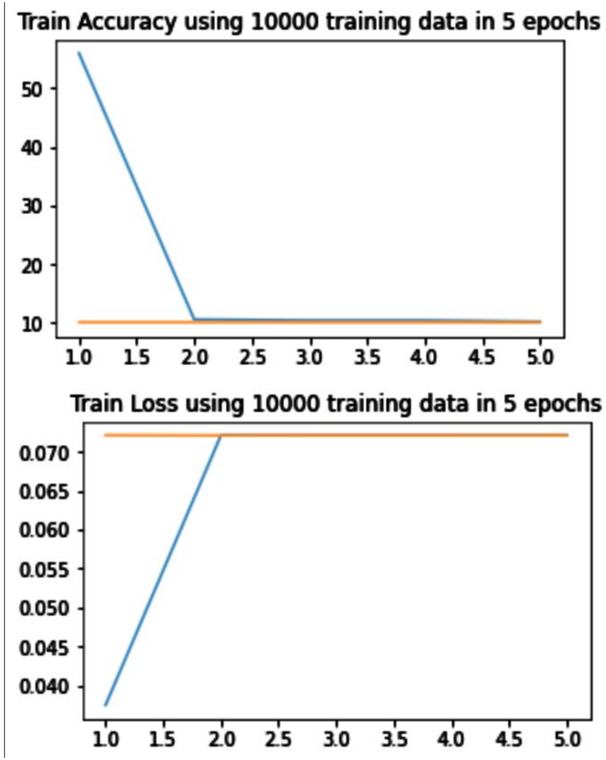
```
Test set: Average loss: 0.0168, Accuracy: 7932/10000 (79.32%), Latency: 110.42 ms
```

Evaluation of FashionMNIST after 5 epochs from pretrained model

Results - Non-pretrained Model

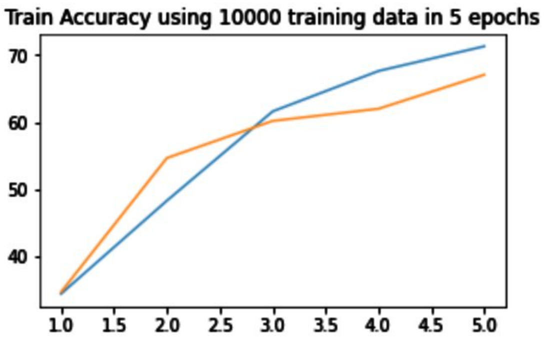


CIFAR10



FashionMNIST

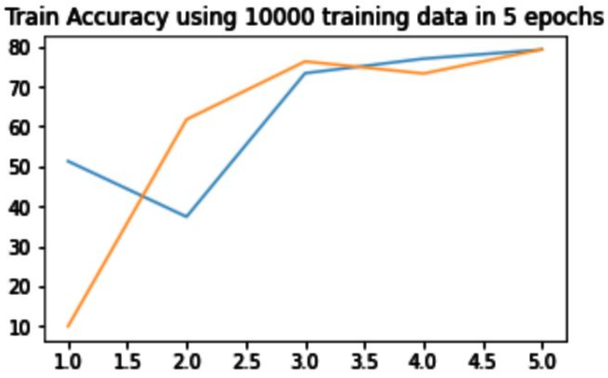
Results - Pretrained Model



Top-1:
67.05%

Latency:
113.39 ms

CIFAR10



Top-1:
79.32%

Latency:
110.42 ms

FashionMNIST

Team Member Duties

- Everyone:
  - Reading and understanding paper
  - Researching open-source repositories
  - Shared Google Colab file to allow parallel progress
  - Presentation + Report
- George and Gilberto
  - Researching Transfer Learning
  - Adjusting model
- Robert:
  - Implementing latency benchmarking
  - Obtaining Results

Challenges

- Deprecated official MNASnet tutorial on Google's Cloud TPU
  - Only works with a billing account
- Imagenet no longer publicly available in 2021
  - Requires free account
  - Deprecates previous tutorials
- Finding well documented repository
- Every dataset is not compatible with the model
  - Requires altering a few layers
- Measuring latency
  - Benchmark can take a very long time to obtain accurate latency for a single model

Conclusion

- What we achieved:
  - Run MNASnetB1 on multiple datasets
  - Configure models to fit desired datasets
  - Implement latency tracking

- What we learned:
  - Public source code can be difficult at times to find
  - Practiced with several cloud platforms and libraries
  - Transfer Learning
  - Code reading skills