GPU Classification of Cifar-10

Neha Joseph & Ashley Herce
TA: Junhao

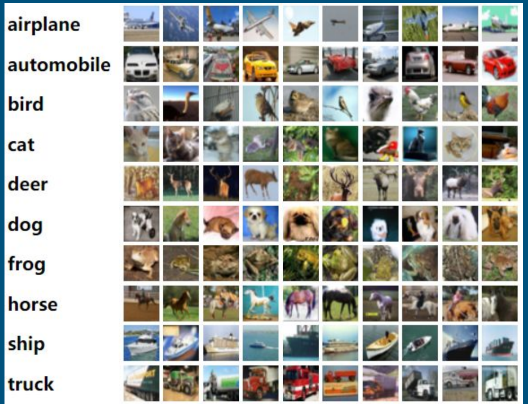
Work partition: Neha: Training, Testing and optimizing AlexNet Ashley: Testing and optimizing ViT-H

Track 2 - AlexNet & ViT-H optimizations on GPU

- We aim to optimize two different models (AlexNet and VIT-H) on the Google Colab GPU(Nvidia Tesla K80 12GB VRAM), and compare the results in change in accuracy and inference latency
  - Our first goal is to recreate similar accuracy results from the published AlexNet and VIT-H models
  - We then aim to replicate similar results while using different optimization techniques such as different types of Quantization and weight pruning
    - For AlexNet we are currently using post training float16 quantization and post training integer quantization
    - For VIT-H we are currently trying to replicate the original model however, we are running into issues as we are restricted by the GPU available and are currently trying to modify our implementation

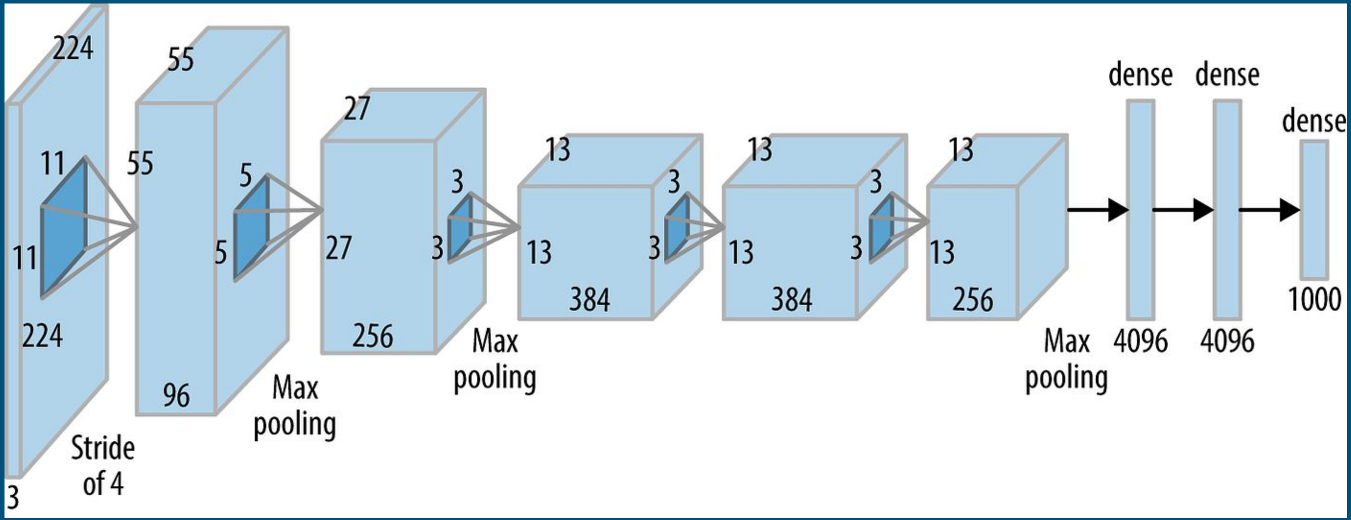
CIFAR10 Dataset

- 10 classes: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships and trucks
- The classes are mutually exclusive and there are no overlaps in between the classes
- 60,000 32x32 colour images in 10 classes split into 50,000 training data, and 10,000 testing data



Design - Alexnet

- 8 layers total
  - 5 convolutional layer (some followed by max pooling layer)
  - 2 fully connected hidden layers
  - 1 fully connected output layer



Design - Alexnet

Layer (type)	Output Shape	Param #
conv2d_20 (Conv2D)	(None, 22, 22, 64)	23296
max_pooling2d_12 (MaxPooling2D)	(None, 10, 10, 64)	0
conv2d_21 (Conv2D)	(None, 10, 10, 192)	307392
max_pooling2d_13 (MaxPooling2D)	(None, 5, 5, 192)	0
conv2d_22 (Conv2D)	(None, 5, 5, 384)	663936
conv2d_23 (Conv2D)	(None, 5, 5, 256)	884992
conv2d_24 (Conv2D)	(None, 5, 5, 256)	590080
max_pooling2d_14 (MaxPooling2D)	(None, 2, 2, 256)	0
flatten_2 (Flatten)	(None, 1024)	0
dense_9 (Dense)	(None, 4096)	4198400
dropout_7 (Dropout)	(None, 4096)	0
dense_10 (Dense)	(None, 4096)	16781312
dropout_8 (Dropout)	(None, 4096)	0
dense_11 (Dense)	(None, 10)	40970

Total params: 23490378 (89.61 MB)
Trainable params: 23490378 (89.61 MB)
Non-trainable params: 0 (0.00 Byte)

Design - ViT-H (Vision Transformer)

Source: https://github.com/lucidrains/vit-pytorch/blob/main/images/vit.gif

- Relies on Transformer architecture (Commonly in NLP)
- Originally designed and tested on larger high resolution datasets (Imagenet21k)
- Computationally expensive due to large size
    - Possible to achieve high accuracy on CIFAR10 but not on Google Colab GPU so far
- Transformer Encoder processes the vectors from patch embedding and the layers within allow the model to attend to specific parts of each patch vector and how it relates to other patches to understand context and spatial relationships
- MLP is a small neural network with hidden layers w/activation function (GELU and dropout)
    - Vector output indicates likelihood of the image belonging to each class

Design - ViT-H (small portion)

Layer (type)	Output Shape	Param #	Connected to
input_layer_12 (InputLayer)	(None, 32, 32, 3)	0	-
data_augmentation (Sequential)	(None, 32, 32, 3)	7	input_layer_12[0][0]
patches_14 (Patches)	(None, 25, 108)	0	data_augmentation[0][…
patch_encoder_6 (PatchEncoder)	(None, 25, 64)	8,576	patches_14[0][0]
layer_normalization_102 (LayerNormalization)	(None, 25, 64)	128	patch_encoder_6[0][0]
multi_head_attention_48 (MultiHeadAttention)	(None, 25, 64)	66,368	layer_normalization_1… layer_normalization_1…
add_96 (Add)	(None, 25, 64)	0	multi_head_attention_… patch_encoder_6[0][0]
layer_normalization_103 (LayerNormalization)	(None, 25, 64)	128	add_96[0][0]
dense_121 (Dense)	(None, 25, 128)	8,320	layer_normalization_1…
dropout_163 (Dropout)	(None, 25, 128)	0	dense_121[0][0]
dense_122 (Dense)	(None, 25, 64)	8,256	dropout_163[0][0]
dropout_164 (Dropout)	(None, 25, 64)	0	dense_122[0][0]
add_97 (Add)	(None, 25, 64)	0	dropout_164[0][0], add_96[0][0]
layer_normalization_104 (LayerNormalization)	(None, 25, 64)	128	add_97[0][0]
multi_head_attention_49 (MultiHeadAttention)	(None, 25, 64)	66,368	layer_normalization_1… layer_normalization_1…
add_98 (Add)	(None, 25, 64)	0	multi_head_attention_… add_97[0][0]
layer_normalization_105 (LayerNormalization)	(None, 25, 64)	128	add_98[0][0]
dense_123 (Dense)	(None, 25, 128)	8,320	layer_normalization_1…
dropout_166 (Dropout)	(None, 25, 128)	0	dense_123[0][0]
dense_124 (Dense)	(None, 25, 64)	8,256	dropout_166[0][0]
dropout_167 (Dropout)	(None, 25, 64)	0	dense_124[0][0]
add_99 (Add)	(None, 25, 64)	0	dropout_167[0][0], add_98[0][0]
layer_normalization_106 (LayerNormalization)	(None, 25, 64)	128	add_99[0][0]
multi_head_attention_50 (MultiHeadAttention)	(None, 25, 64)	66,368	layer_normalization_1… layer_normalization_1…
add_100 (Add)	(None, 25, 64)	0	multi_head_attention_… add_99[0][0]
layer_normalization_107 (LayerNormalization)	(None, 25, 64)	128	add_100[0][0]
dense_125 (Dense)	(None, 25, 128)	8,320	layer_normalization_1…
dropout_169 (Dropout)	(None, 25, 128)	0	dense_125[0][0]
dense_126 (Dense)	(None, 25, 64)	8,256	dropout_169[0][0]
dropout_170 (Dropout)	(None, 25, 64)	0	dense_126[0][0]
add_101 (Add)	(None, 25, 64)	0	dropout_170[0][0], add_100[0][0]
layer_normalization_108 (LayerNormalization)	(None, 25, 64)	128	add_101[0][0]
multi_head_attention_51 (MultiHeadAttention)	(None, 25, 64)	66,368	layer_normalization_1… layer_normalization_1…
add_102 (Add)	(None, 25, 64)	0	multi_head_attention_… add_101[0][0]
layer_normalization_109 (LayerNormalization)	(None, 25, 64)	128	add_102[0][0]
dense_127 (Dense)	(None, 25, 128)	8,320	layer_normalization_1…
dropout_172 (Dropout)	(None, 25, 128)	0	dense_127[0][0]
dense_128 (Dense)	(None, 25, 64)	8,256	dropout_172[0][0]

Results - Model accuracy on Test Data

- Current results for AlexNet

```
Top 1 accuracy:  0.5386
Top 5 accuracy:  0.8949
72.67679471700467 ms
```

- Current results for ViT-H

```
Test accuracy: 18.78%
Test top 5 accuracy: 76.01%
time: 1min 26s (started: 2024-04-24 23:03:39 +00:00)
```

*Accuracy after training on ¼ the Training Data

Next Steps: Work on optimizing latency of both models and comparing results of accuracy, inference times of both models and detailing the tradeoffs and benefits of each model