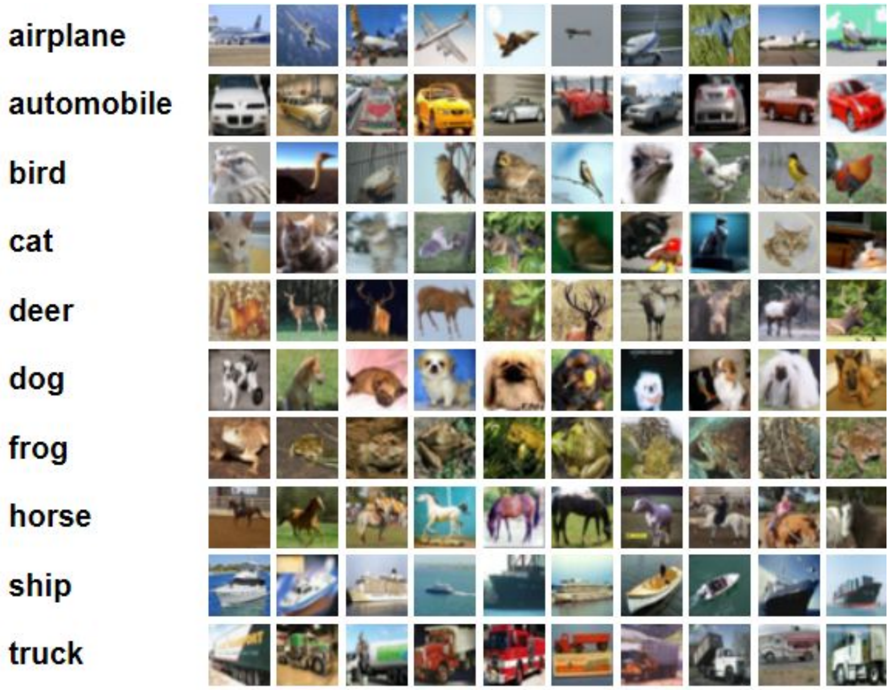
Vision Transformers

Contents

CIFAR-10 Dataset

- The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class.
- There are 50000 training images and 10000 test images.
- Example of 10 images for each of the 10 classes.



Vision Transformer

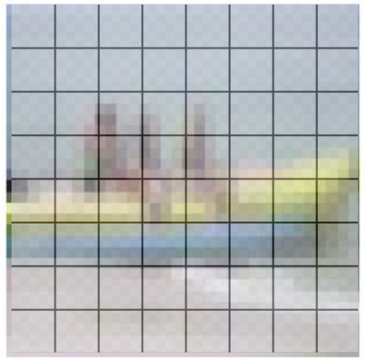
- Transformer architecture has become the de-facto standard in NLP tasks.
- Through the paper, “AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE” authors propose a method to apply Transformers to Vision tasks with very little modification on the transformers for text. [Link to paper]
- In CNNs, locality, two-dimensional neighborhood structure, and translation equivariance are baked into each layer throughout the whole model.
- Transformers lack some of these above inductive biases inherent to CNNs and therefore do not generalize well when trained on insufficient amounts of data.
- However, the picture changes if the models are trained on larger datasets (14M-300M images) and authors propose that large scale training trumps inductive bias.

Vision Transformer - Illustration

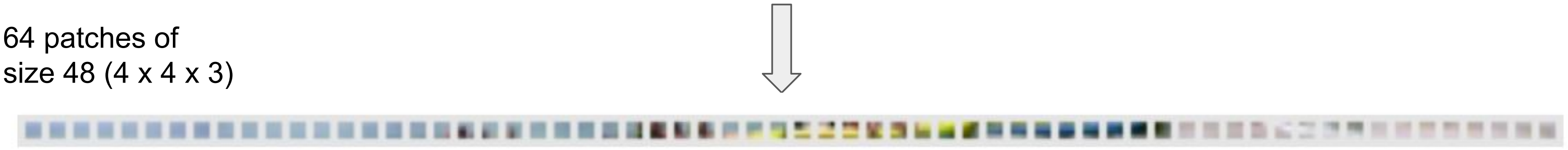
Vision Transformer - Image to Patch

(32 x 32 x 3) image
Divide into patch_size = 4
=> 32/4 x 32/4 x 3
=> 64 x (4 x 4 x 3)
=> 64 x 48 for each image

Image example of the CIFAR10 dataset



64 patches of
size 48 (4 x 4 x 3)



Vision Transformer - Linear Proj, CLS token and POS embeddings

Besides the Multi-Headed Attention layer, Vision Transformer consists of:

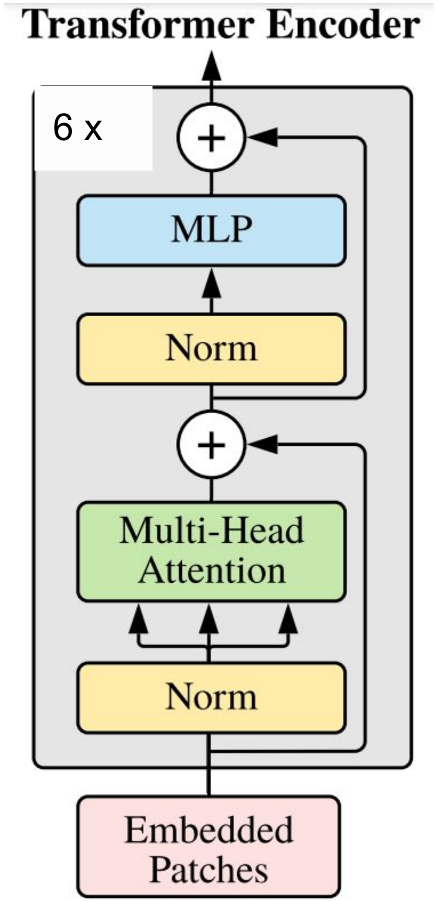
- A **linear projection layer** that maps the input patches to a feature vector of larger size. It is implemented by a simple linear layer that takes each M X M patch independently as input.
- A **classification token** that is added to the input sequence. We will use the output feature vector of the classification token (CLS token in short) for determining the classification prediction.
- **Learnable positional encodings** that are added to the tokens before being processed by the Transformer. Those are needed to learn position-dependent information, and convert the set to a sequence. Since we usually work with a fixed resolution, we can learn the positional encodings instead of having the pattern of sine and cosine functions.
- Finally, an **MLP head** that takes the output feature vector of the CLS token, and maps it to a classification prediction. This is usually implemented by a small feed-forward network or even a single linear layer.

Vision Transformer - Training

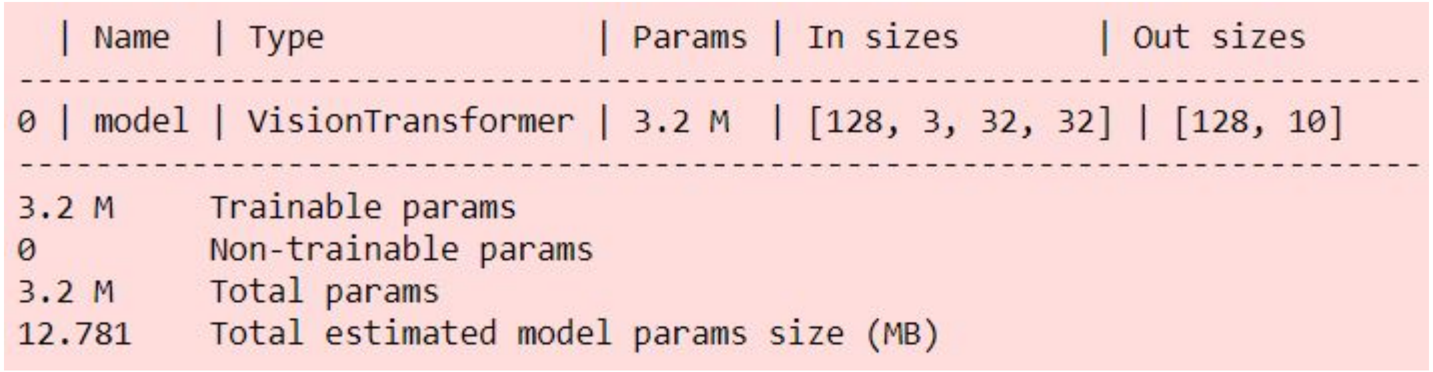
Hyperparameters for the Vision Transformer

- Feature length input image projections **'embed_dim'**: 256
- Hidden dimension length for attention vectors **'hidden_dim'**: 512,
- Number of attention heads **'num_heads'**: 8,
- Number of stacked transformers layers **'num_layers'**: 6,
- Patch size of the input image patch **'patch_size'**: 4,
- Number of channels in the input images **'num_channels'**: 3,
- Number of patches generated from single input image **'num_patches'**: 64,
- Number of output classes in CIFAR-10 **'num_classes'**: 10,
- Dropout for regularization **'dropout'**: 0.2
- Learning rate **‘lr’**: 3e-4
- **‘batch_size’** : 128

Number of epochs the model is trained: 180



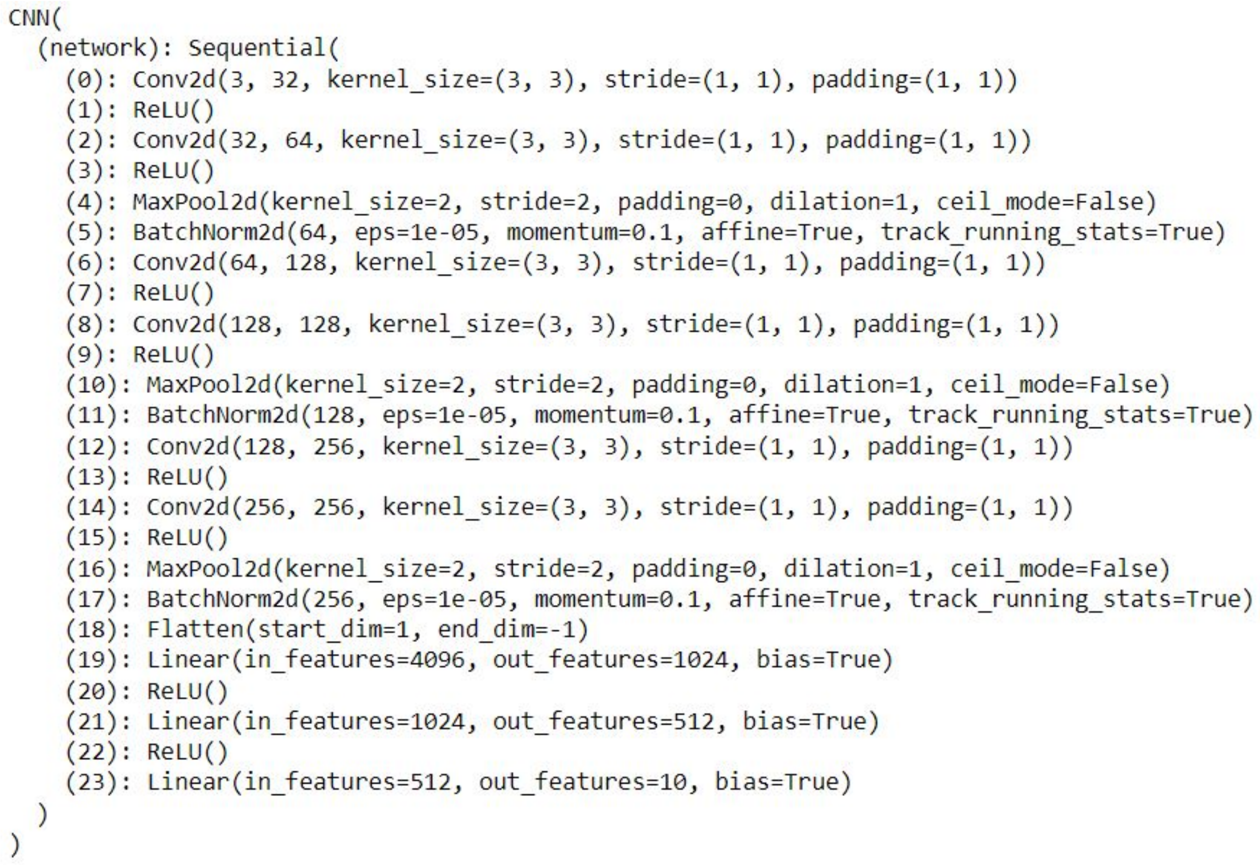
Vision Transformer - Training



```
  | Name  | Type              | Params | In sizes         | Out sizes
----------------------------------------------------------------------------
0 | model | VisionTransformer | 3.2 M  | [128, 3, 32, 32] | [128, 10]
----------------------------------------------------------------------------
3.2 M     Trainable params
0         Non-trainable params
3.2 M     Total params
12.781    Total estimated model params size (MB)
```

Model size and number of parameters

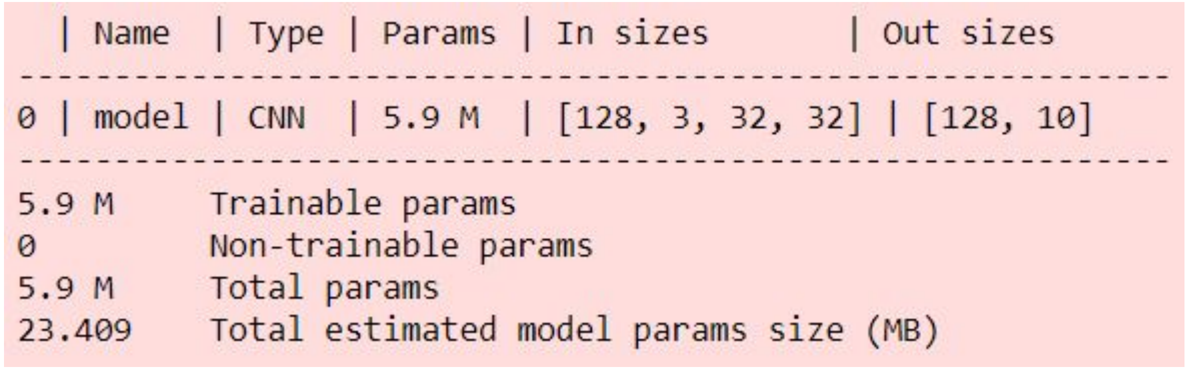
CNN - Training



```
CNN(
  (network): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU()
    (8): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU()
    (10): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (11): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (12): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU()
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU()
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (17): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (18): Flatten(start_dim=1, end_dim=-1)
    (19): Linear(in_features=4096, out_features=1024, bias=True)
    (20): ReLU()
    (21): Linear(in_features=1024, out_features=512, bias=True)
    (22): ReLU()
    (23): Linear(in_features=512, out_features=10, bias=True)
  )
)
```

Training details:
Learning_rate: 3e-4
Batch_size: 128

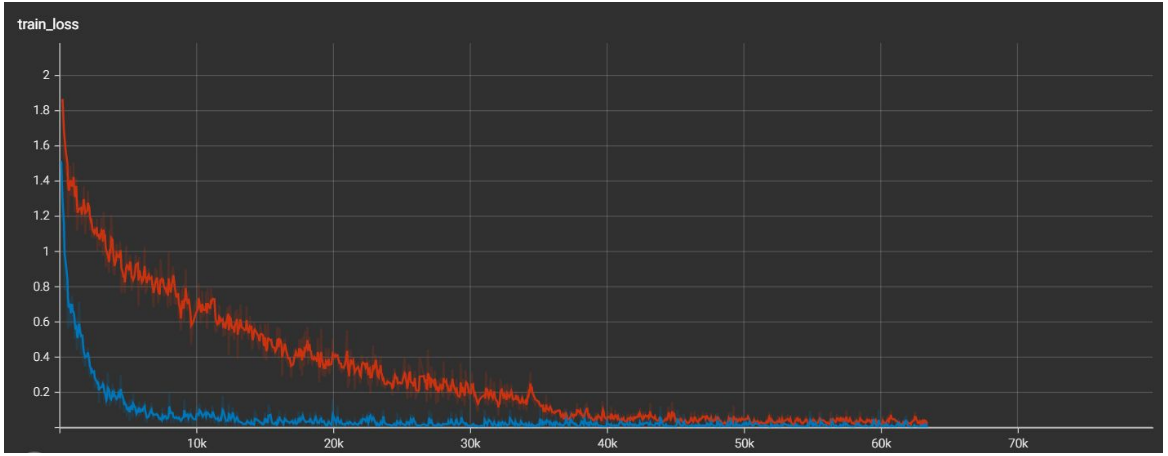
CNN

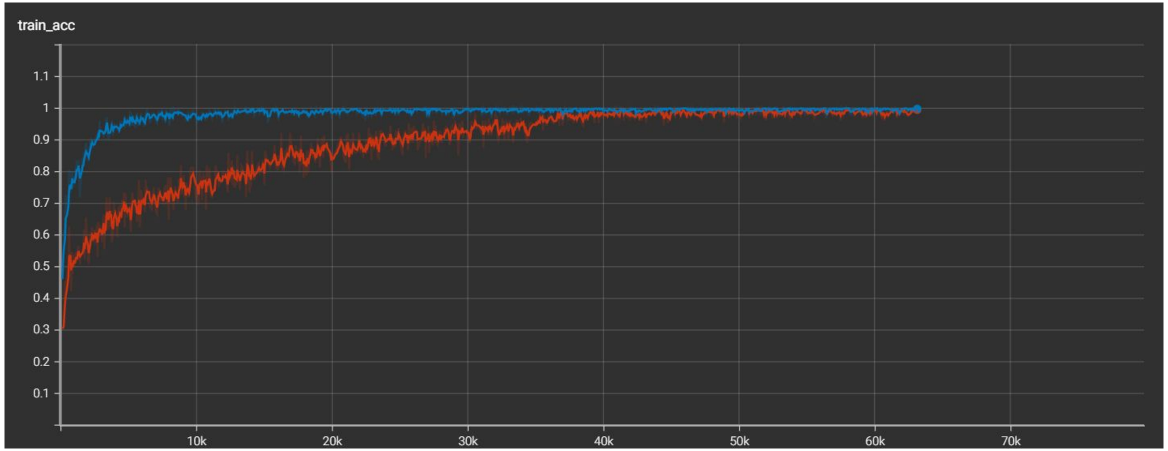


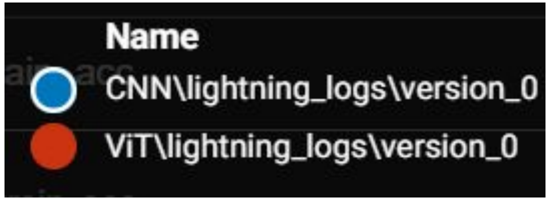
```
  | Name  | Type | Params | In sizes         | Out sizes
----------------------------------------------------------------
0 | model | CNN  | 5.9 M  | [128, 3, 32, 32] | [128, 10]
----------------------------------------------------------------
5.9 M     Trainable params
0         Non-trainable params
5.9 M     Total params
23.409    Total estimated model params size (MB)
```

Model size and number of parameters

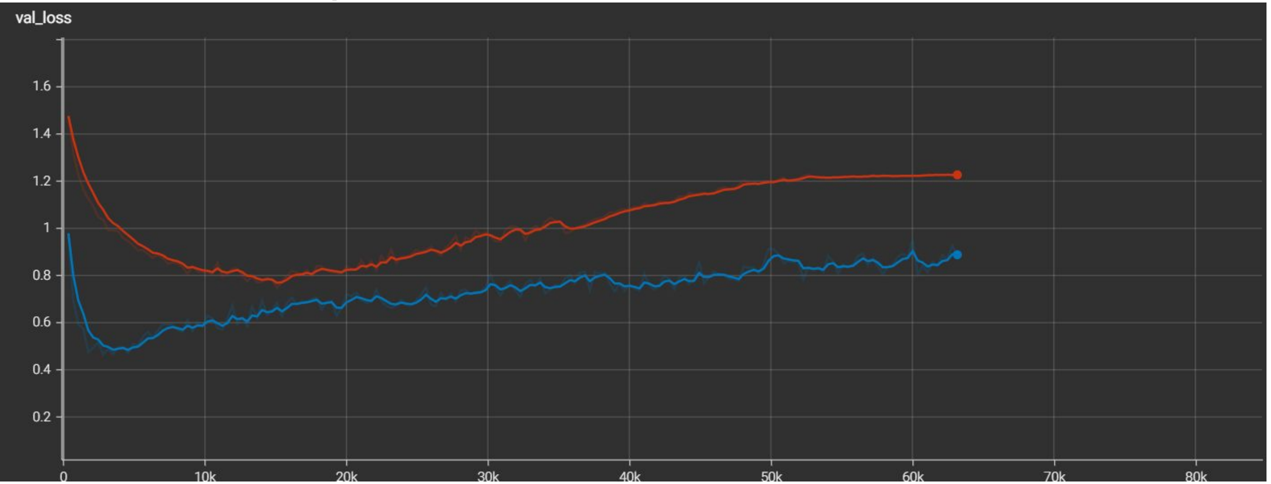
Training comparisons: ViT vs. CNN

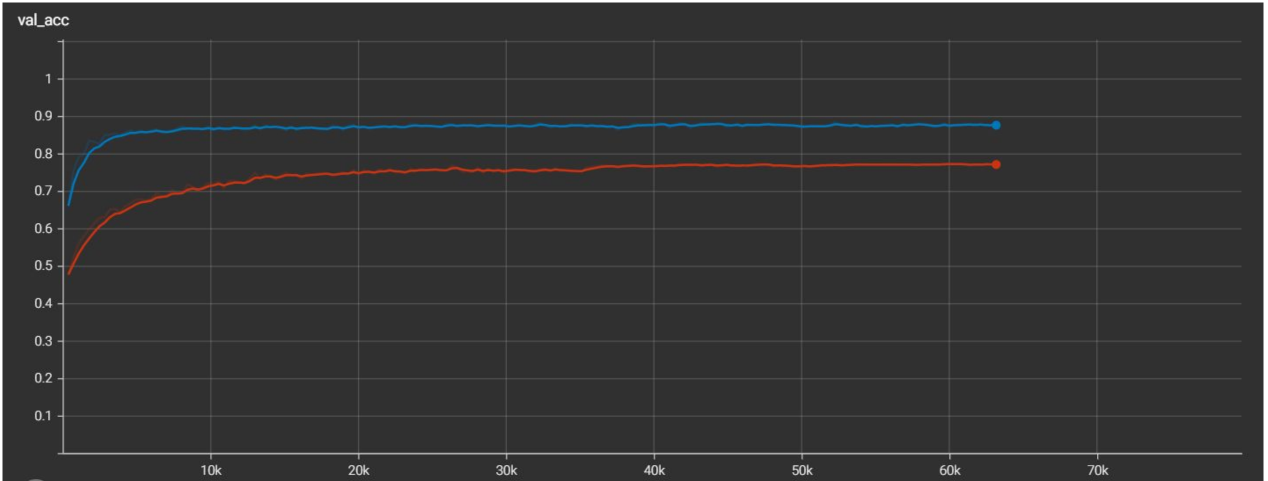


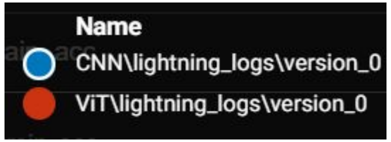



Name
CNN\lightning_logs\version_0
ViT\lightning_logs\version_0

Validation comparisons: ViT vs. CNN







Test loss and accuracy

Finally, comparing the accuracy and loss on the test set of CIFAR-10.

Model	Accuracy	Loss
CNN	89%	0.86
ViT	77%	1.28

- Overall CNN model achieves better test accuracy and also test loss.

ViT vs. CNN - Observations

- From the training curves, we can observe that CNN **fits the data very quickly** and achieves higher accuracy is less number of training steps when compared to Vision Transformer.
    - Both are able to achieve achieve the accuracy of 100% and a loss ~ 0 on training data

- From the validation curves, it is clear that CNN performs well achieving validation accuracy of 88% where as ViT struggles with accuracy of 77%

- This seemingly discouraging outcome for the ViT may be expected: Transformers lack some of the inductive biases inherent to CNNs, such as translation equivariance and locality, and therefore do not generalize well when trained on insufficient amounts of data (such as CIFAR-10 with only 45K training images).

- However, the picture changes if the models are trained on larger datasets (14M-300M images). Authors propose that in this case, large scale training trumps inductive bias and provide SOTA results on benchmarks with larger models on large datasets.

Thank you!