## **PyTorch Implementation of Classic Deep Learning Papers**

## **Key Learnings & Insights**

### **ResNet-18 (CIFAR-10)**

* Implementing residual blocks required careful handling of identity vs. projection shortcuts to align dimensions.
* Using global average pooling instead of a dense flatten layer reduced parameters and helped generalization.
* Training curves showed steady improvement; by epoch 9, training accuracy reached -67%, with test accuracy -67.5%. Continued training is expected to cross 80% as reported in the paper.
* Visuals such as loss/accuracy curves (see curves\_cls.png) revealed healthy convergence.

### **Transformer (Toy Translation)**

* Implementing multi-head attention and positional encodings from scratch clarified how sequence alignment works without recurrence.
* Attention heatmaps revealed interpretable diagonal alignment between input and reversed-output tokens.

## **Challenges & Resolutions**

* Residual connections: mismatched dimensions between input and output channels were solved using 1×1 convolutions (projection shortcuts).
* Training stability: initially the Transformer diverged; fixed with learning rate warm-up and gradient clipping.
* Faced hardware challenges as the device is not compatible to train the model from scratch on resnet.

## **Practice Exercises**

* Due to hardware limitations, the complete training process could not be executed on the local machine. Only partial training was conducted to demonstrate the functionality of the model
* Wrote a mini residual block test script (forward pass on random tensors).
* Implemented a standalone scaled dot-product attention function and validated against PyTorch’s outputs.
* Debugged causal masks with a toy sequence [1,2,3,4] ensuring no future token leakage.

## **Deliverables**

### **Source Code**

* **ResNet** (code/resnet.py, code/train\_resnet.py)  
  + Residual block & ResNet-18 architecture from scratch.
  + CIFAR-10 dataloader with augmentation.
  + Training loop, validation evaluation.
* **Transformer** (code/transformer.py, code/train\_transformer.py)  
  + Embeddings, sinusoidal position encoding, encoder/decoder stacks, attention, feed-forward layers, masking.

## **Sources Consulted**

* **ResNet**:  
  + He, K., et al. (2015). *Deep Residual Learning for Image Recognition*. [arXiv:1512.03385](https://arxiv.org/abs/1512.03385)
  + PyTorch documentation for torch.nn.Conv2d, torch.nn.BatchNorm2d, torch.nn.Linear, torch.utils.data.DataLoader
  + Tutorials and discussions from GeeksforGeeks on [loading CIFAR-10](https://www.geeksforgeeks.org/python/how-to-load-cifar10-dataset-in-pytorch/)
  + scikit-learn docs for [confusion\_matrix](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html)
  + Selvaraju et al. (2016). *Grad-CAM*. [arXiv:1610.02391](https://arxiv.org/abs/1610.02391)
* **Transformer**:  
  + Vaswani, A., et al. (2017). *Attention Is All You Need*. [arXiv:1706.03762](https://arxiv.org/abs/1706.03762)
  + Huang, H. *Sequence-to-Sequence Modeling with nn.Transformer and TorchText*. [Tutorial](https://h-huang.github.io/tutorials/beginner/transformer_tutorial.html)
  + NLTK documentation for BLEU score implementation ([Papineni et al., 2002](https://aclanthology.org/P02-1040/))