EE 8520 - Homework 1

CIFAR-10 Classification

Due: September 25, 2025   **Template:** click here

! Please follow the "rules for every HW" listed in the syllabus

Programming Exercise: In this assignment, you will implement several different neural network models (MLP, CNN, and ViT) and apply them to the CIFAR-10 classification task. You are provided with the skeleton codes `hw1.py` and `vit.py` to help you get started.

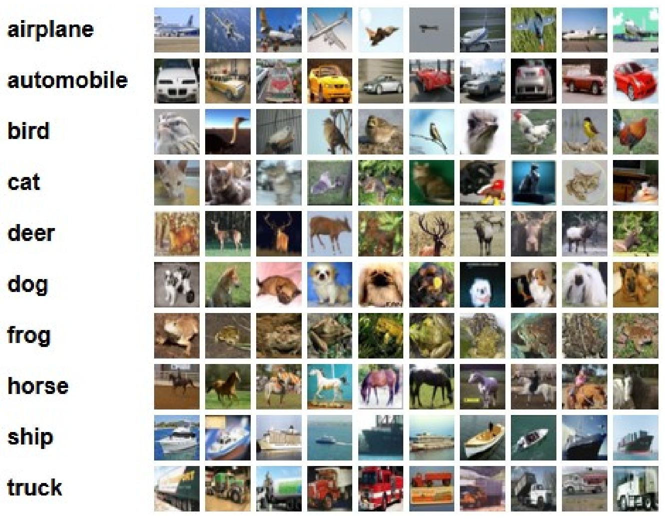


Figure 1: CIFAR-10 dataset.

Step 1 (15 points): Multi-Layer Perceptron (MLP)

Your first task is to implement a multi-layer perceptron (Goodfellow et al., 2016) for CIFAR-10 classification. You are given a skeleton code `hw1.py`. Make sure to check lines with `#TODO` flag in order to complete the code.

```
# CODE SNIPPET FOR SINGLE LINEAR LAYER
class MLP_classifier(nn.Module):
    def __init__(self, in_channels, num_classes):
        super().__init__()
        self.flatten = ...
        self.fc = ...
    def forward(self, x):
        y = ... # flatten your input
        y = ... # apply the linear classifier
        return y
```

For this step, your goals are:

- Implement a **single linear layer (i.e., a linear classifier)** that maps flattened images to class logits. Use the given code snippet as a backbone.

- Extend your model into a **multi-layer perceptron (MLP)** by stacking multiple linear layers (the number is up to you). Do you see any performance increase? *Hint: Do not forget to report the hidden layer dimensions you chose and why you chose it.*
- Now, try adding non-linear activation functions between linear layers. Discuss whether the performance has improved, and explain why or why not.
- Experiment with different hyperparameters (*e.g.*, learning rates, batch sizes, number of epochs, optimizers, loss functions). **Justify your reasoning for each**: why did you try it, what worked better, what did not?
- For each design choice, report the validation accuracy curve (3 curves), loss curve (3 curves), and final test accuracy (3 final accuracies). Also report the confusion matrices for each classification task (3 confusion matrices).

In the skeleton code, you are also given an example visualization code that plots the image, its true label and its predicted one. Try to include some examples in your report.

Step 2 (15 points): Convolutional Neural Networks (CNNs)

In this step, you will be implementing a CNN (Krizhevsky et al., 2017) for CIFAR-10 classification. Again, use the given `hw1.py` file as a skeleton. Your goals are:

- Design and implement a custom CNN architecture that uses convolutional layers, non-linear activation functions, pooling layers, and fully connected layers. Explore different design choices such as number of channels (*i.e.*, number of filters) $\in \{32, 64, 128, 256\}$, kernel sizes $\in \{3, 5, 7\}$, pooling strategies, depth, use of dropout. The specific design choices are up to you, but make sure to **report your observations** and **explain the reasoning** behind your selections. *Hint: To keep experiments manageable, you may use the same kernel size across all convolutional layers when testing different values. Increasing the number of channels after each pooling operation may also help.*
- Again, try multiple hyperparameters (learning rate, batch size, etc.) to achieve the best performance and include your findings in the report.
- Reflect on why your CNN cannot achieve very high accuracy (above 90%) on CIFAR-10. Discuss the limitations of your model design, the dataset, and the training setup that might prevent reaching top-level performance. Which classes do you think are hardest to distinguish? *Hint: Use the confusion matrix.*
- Compare the CNN performance with the best-performing MLP from Step 1. What improves, and why? Which one do you think is faster or less memory consuming? Are there any advantages of using convolutional layers? Explain your reasoning.

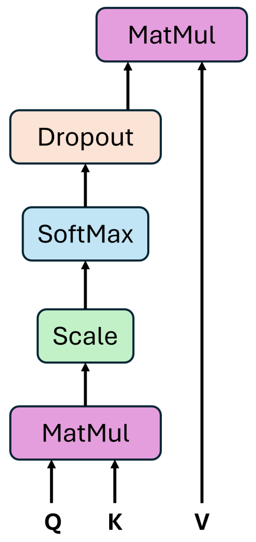
For this implementation, **try your best to reach a test accuracy over 75%**. Similar to step-1, report the validation accuracy curve, loss curve, final test accuracy (only for your best-performing CNN model) along with the confusion matrix (1 confusion matrix).

Step 3 (20 points): Vision Transformer (ViT)

Finally, you will implement a Vision Transformer (Dosovitskiy et al., 2021) for CIFAR-10 classification. A starting code (`vit.py`) is provided to you. Your main task is to implement the **Self-Attention** module using the hints provided via `#TODO` flags.

Your goals are:

- Fill in the **Self-Attention** block in `vit.py`, following the given figure on the right. Notice the difference between this figure and the figure you have seen in the class (Lecture 3 - Attention, page 29). Comment on why we do not need that optional masking operation for CIFAR-10 classification using ViTs.
- Understand and explain what queries (Q), keys (K), and values (V) represent in self-attention, how they are computed from the input embeddings, and describe step by step how they are combined to produce the attention output.
- Train your ViT on CIFAR-10 and compare its performance with your CNN and MLP. Discuss any challenges you observe (e.g., training time, convergence behavior, accuracy).
- Again, try multiple hyperparameters (learning rate, batch size, etc.) to achieve the best performance and include your findings in the report.



Similar to previous steps, report the validation accuracy curve, loss curve, final test accuracy (only for your best-performing ViT model) along with the confusion matrix (1 confusion matrix).

Deliverables: Upload 2 separate files: **a PDF report** (`lastname_EE8520_F25.pdf`) and **a zipped file (or a single .py or .ipynb script)** (`lastname_EE8520_F25.zip(.py/.ipynb)`), containing your codes. Submit only the implementations of your best-performing models for each part (MLP, CNN, and ViT). Ideally, you should only need to turn in `hw1.py` and `vit.py`; all models can be implemented in these two files, so there is no need to create separate versions of `hw1.py`.

References

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.