# Training Dynamics Mini Project

This mini project will be less intense than the previous ones. In this mini project, you will train a neural network to classify a dataset of your choice. Chose any image dataset you like, but it a task that allows for optimization. You could use MNIST, or look on Kaggle for a dataset that interests you. Use convolutional layers and a few dense layers.

## Requirements

1. You must use pure JAX.
2. You must use `jit`, `grad`, and `vmap`.
3. You must train using `jax.lax.scan` (use `ys` to accumulate metrics).
4. You must use `optax` for optimization.
5. You must use dropout for regularization (use `rngs` as `xs` for the `scan`).
6. Choose one of the convolutional layers and animate the channels throughout training.

## Deliverables

The output of this project will be:

1. Some beautiful and concise array programming code.
2. A mesmerizing animation of the channels of one of the convolutional layers.

## Notes on `jax.lax.scan`

A scan is a functional programming concept that is used to accumulate state. Conceptually, `lax.scan` can be thought of as:

```
def scan(f, init, xs, length=None):
  if xs is None:
    xs = [None] * length
  carry = init
  ys = []
  for x in xs:
    carry, y = f(carry, x)
    ys.append(y)
  return carry, np.stack(ys)
```

Instead of that, you get to write:

```
params, ys = lax.scan(f, init, xs)
```

It is also highly optimized and can be used to accumulate gradients, metrics, etc. I tend to have the state hold my parameters and the ys return my metrics. But, `scan` is very flexible and can be used in many ways.

## Deadline

The deadline is October 13th, 2024.