OML Report

Task 1

- *Divyansh Rastogi [2019464]*
- *Rupanshu Yadav [2019475]*

Task

Image Classification

Dataset

CIFAR-10 [Link]

Environment Specifications

Experiments were conducted on Google Colab platform with a GPU runtime. The following are the CPU & GPU details:

```
1 !lscpu
```

```
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              2
On-line CPU(s) list: 0,1
Thread(s) per core:  2
Core(s) per socket:  1
Socket(s):           1
NUMA node(s):        1
Vendor ID:           GenuineIntel
CPU family:          6
Model:               63
Model name:          Intel(R) Xeon(R) CPU @ 2.30GHz
Stepping:            0
CPU MHz:             2299.998
BogoMIPS:            4599.99
Hypervisor vendor:   KVM
Virtualization type: full
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            46080K
NUMA node0 CPU(s):   0,1
```

```
1 !nvidia-smi -L
```

```
GPU 0: Tesla K80 (UUID: GPU-89bda309-ea8f-dc05-c696-c9c74c115310)
```

Model

```
CNN(
  (conv): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1))
    (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (2): ReLU()
  )
  (lin): Sequential(
    (0): Linear(in_features=7200, out_features=600, bias=True)
    (1): ReLU()
    (2): Linear(in_features=600, out_features=120, bias=True)
    (3): ReLU()
    (4): Linear(in_features=120, out_features=10, bias=True)
  )
  (softmax): LogSoftmax(dim=1)
)
```

Loss Criterion

Negative Log Likelihood Loss < `nn.NLLLoss()` >

Experiments

The dataset was subsampled and divided into batch sizes of 64. The model was run for 100 epochs. Each optimizer was run with its default/most commonly used hyperparameters and with its best-obtained parameters from hyperparameter tuning.

Hyperparameter tuning:

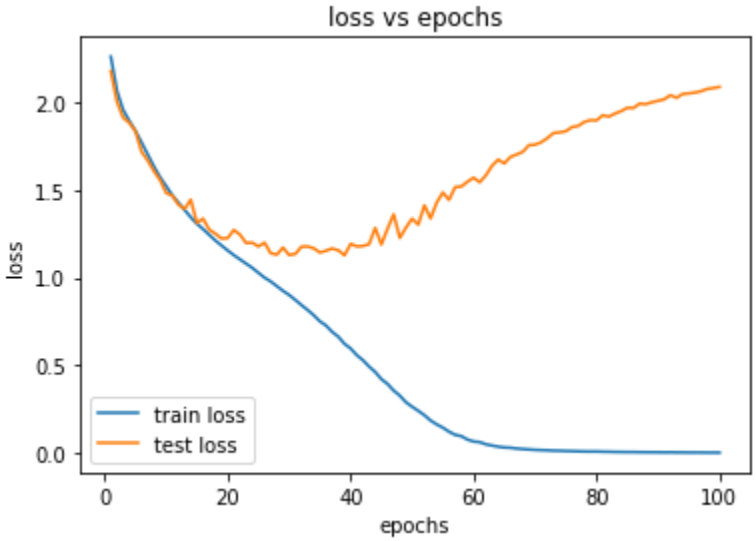
For a set of hyperparameters, various trials with different optimizer settings are trained based on randomly-sampled or exhaustive hyperparameters up to a certain number of epochs (dependent on learning rate). Finally, the best set of hyperparameters is chosen, which corresponds to the minimum testing loss after training.

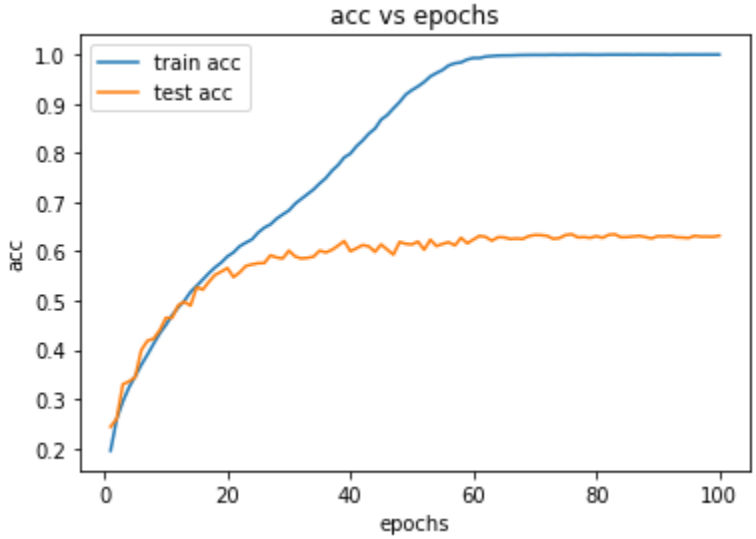
Each optimizer is implemented in a different ipynb-notebook of its own. The following optimizers were run:

➔ **SGD**

Default hyperparameters:

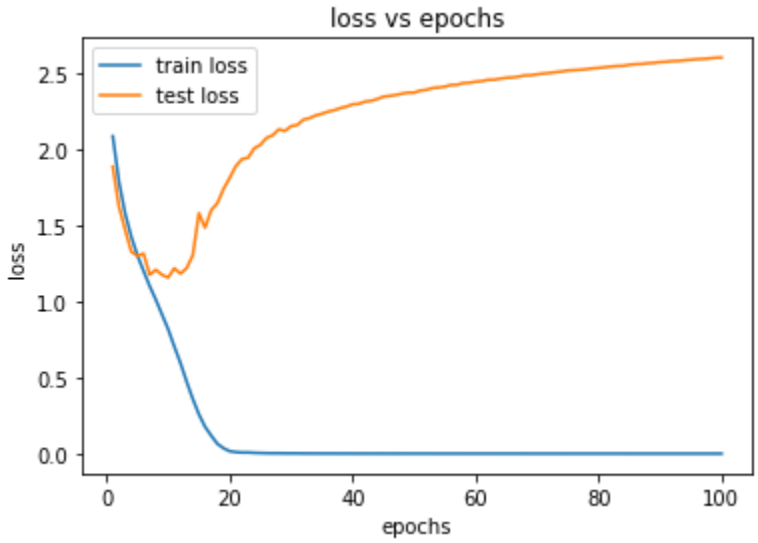
```
{lr: 0.01}
```

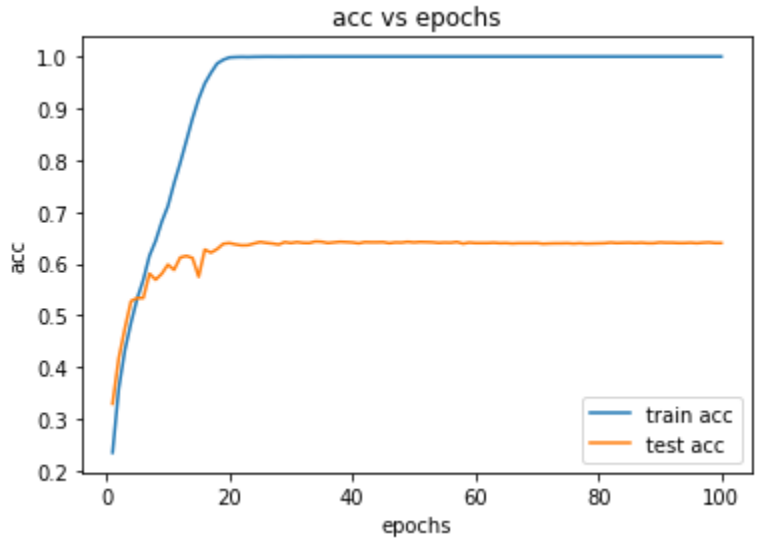




Optimal hyperparameters:

```
{lr: 0.05}
```





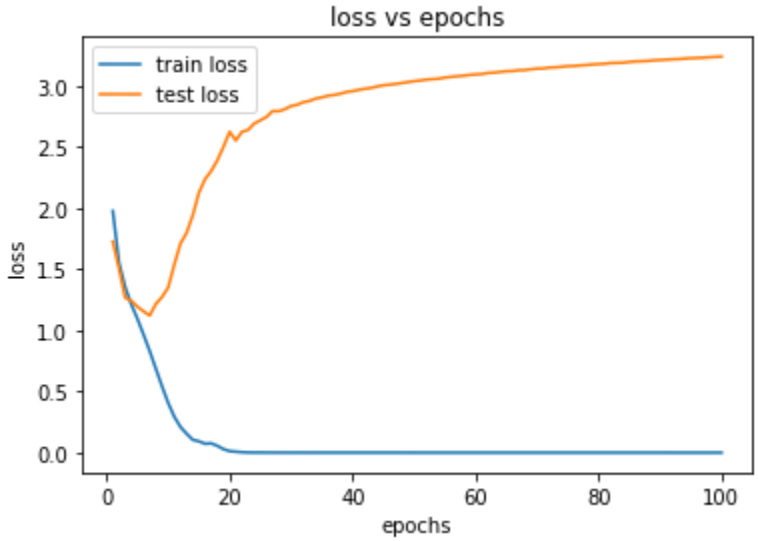
Sampling domain:

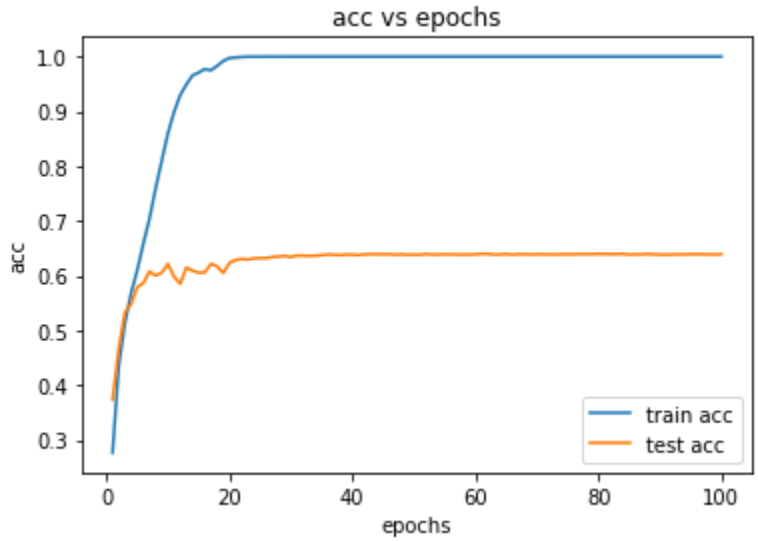
```
TRIALS = 6

HYPERPARAMS = {
    'lr': [0.1, 0.05, 0.01, 0.005, 0.001, 0.0001]
}
```

➔ **Momentum SGD**

Default hyperparameters:

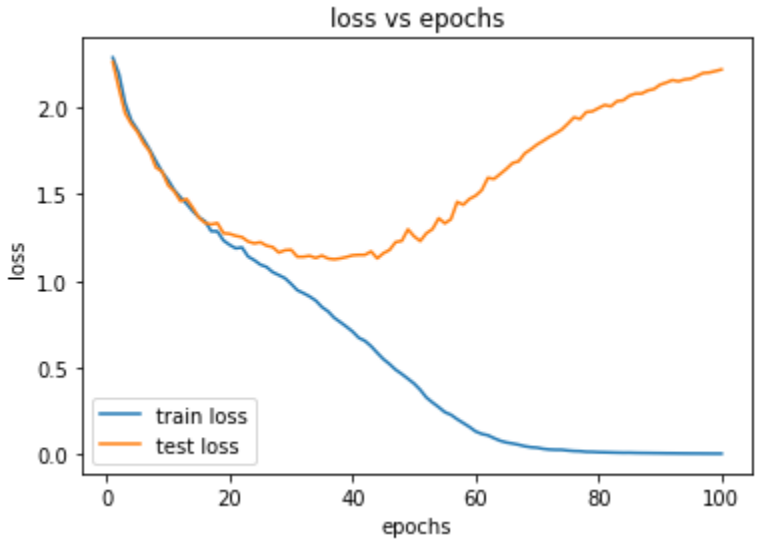
```
{lr: 0.01, momentum: 0.9}
```

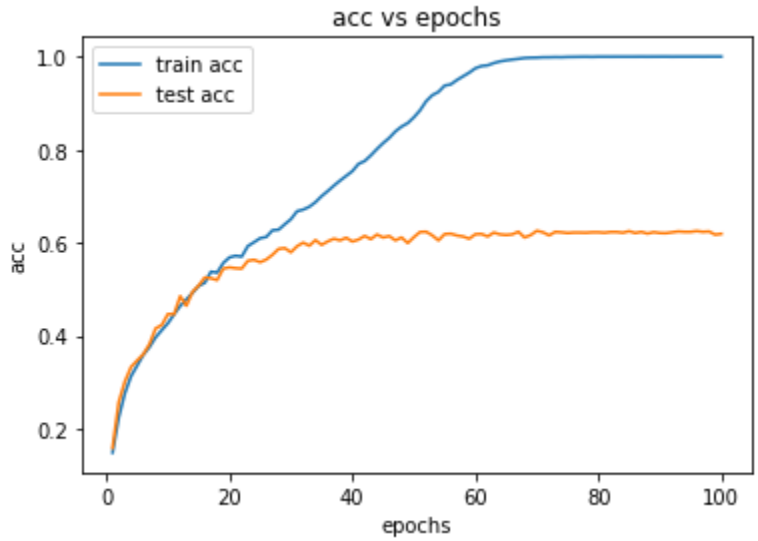




Optimal hyperparameters:

```
{lr: 0.0001, momentum: 0.99}
```





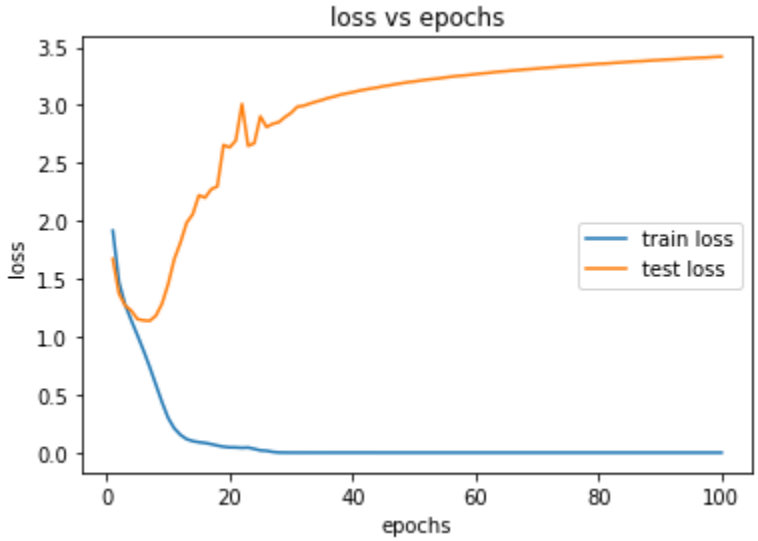
Sampling domain:

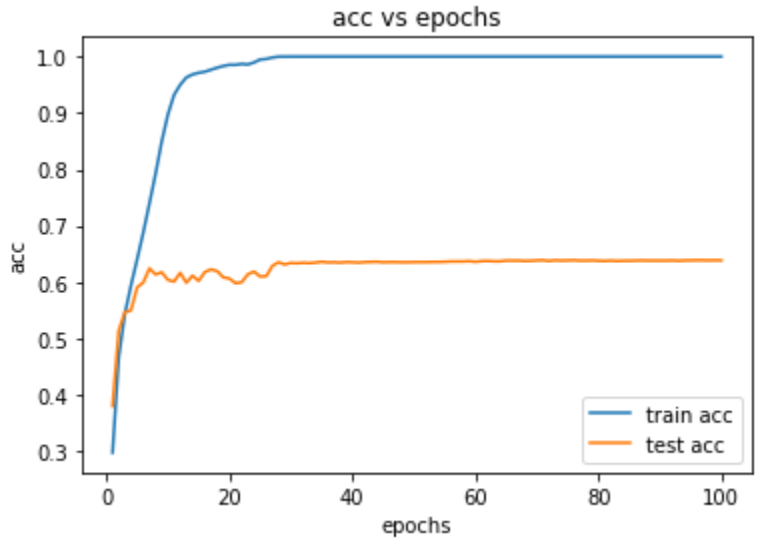
```
TRIALS = 10

HYPERPARAMS = {
    'lr': [0.1, 0.05, 0.01, 0.005, 0.001, 0.0001],
    'momentum': [0.3, 0.5, 0.7, 0.9, 0.99]

}
```

➔ **NAG SGD**

Default hyperparameters:

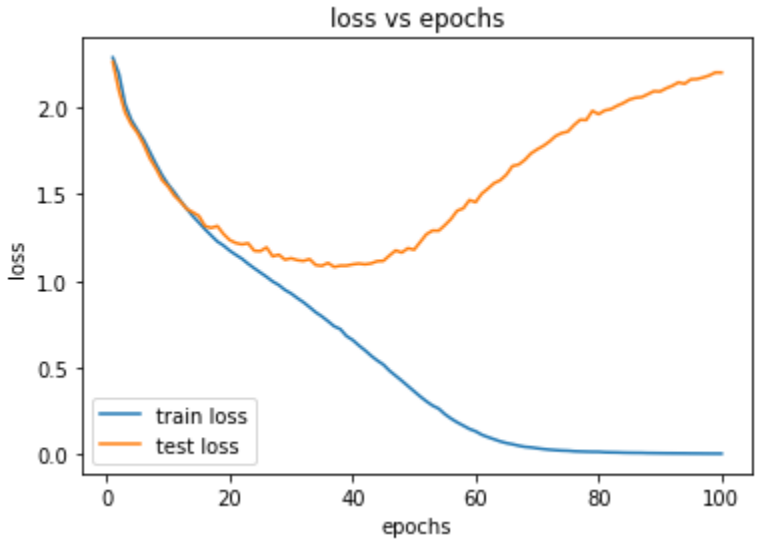
```
{lr: 0.01, momentum: 0.9}
```

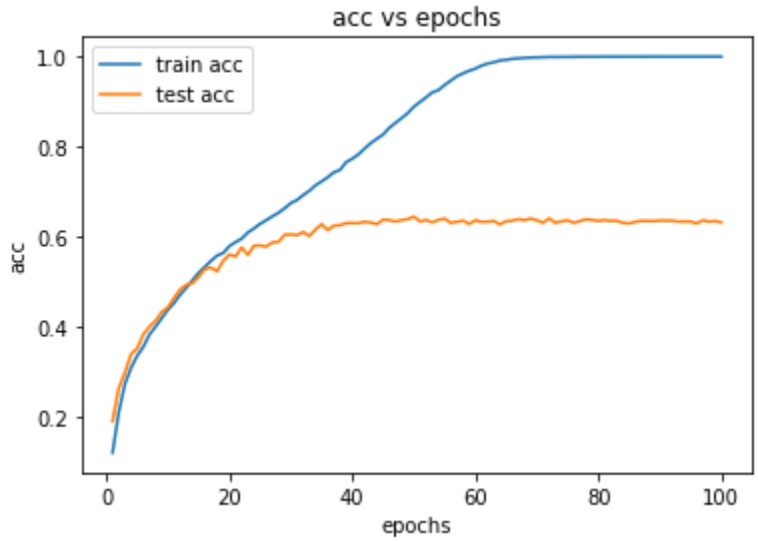




Optimal hyperparameters:

```
{lr: 0.0001, momentum: 0.99}
```





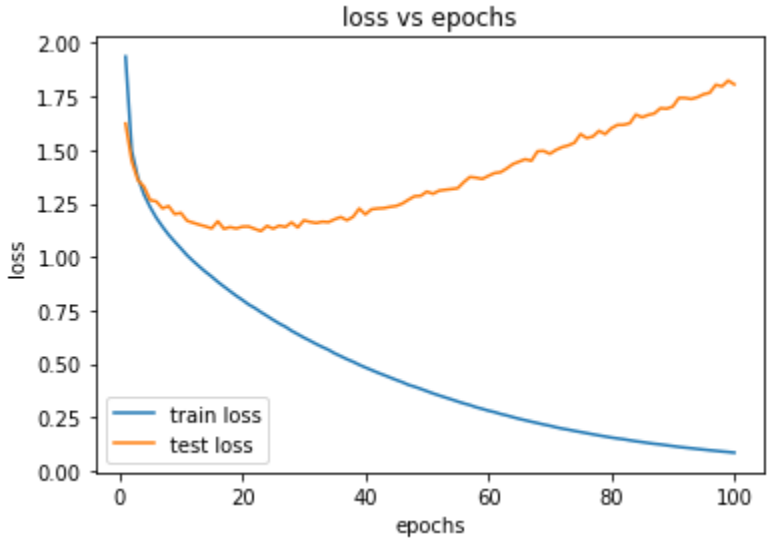
Sampling domain:

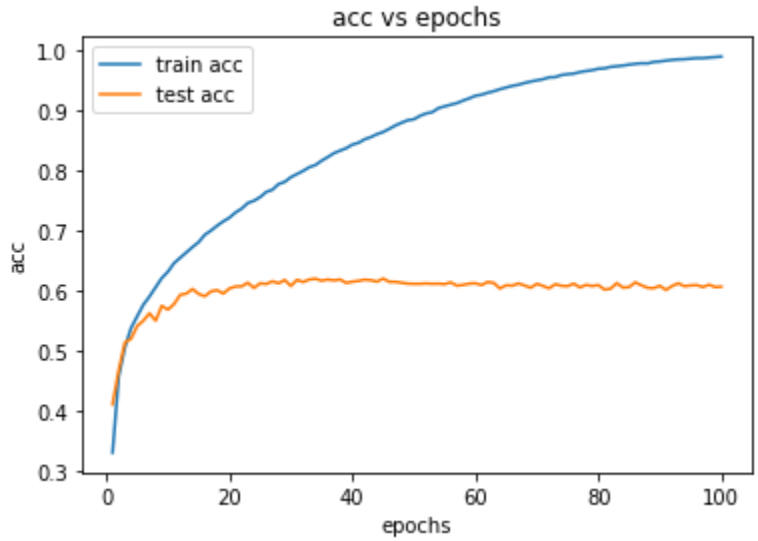
```
TRIALS = 10

HYPERPARAMS = {
    'lr': [0.1, 0.05, 0.01, 0.005, 0.001, 0.0001],
    'momentum': [0.3, 0.5, 0.7, 0.9, 0.99]
}
```

➔ **AdaGrad**

Default hyperparameters:

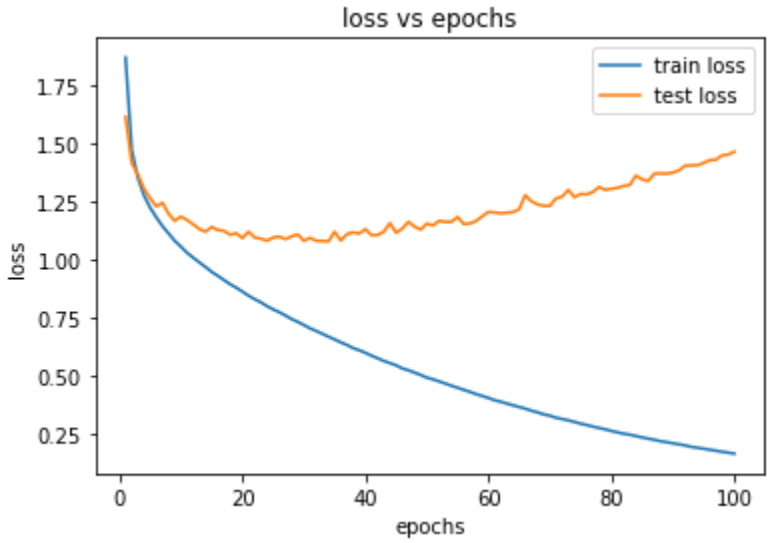
```
{lr: 0.01, eps: 1e-8}
```

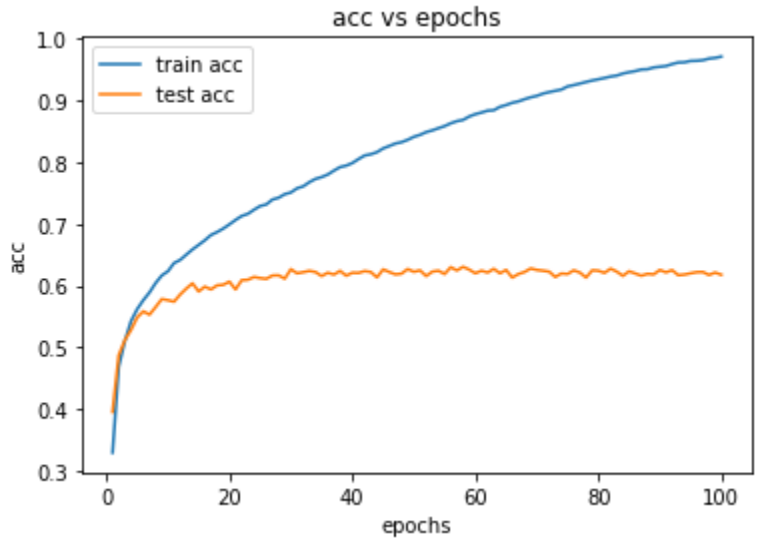




Optimal hyperparameters:

```
{eps: 1e-09, lr: 0.01, weight_decay: 0.0001}
```





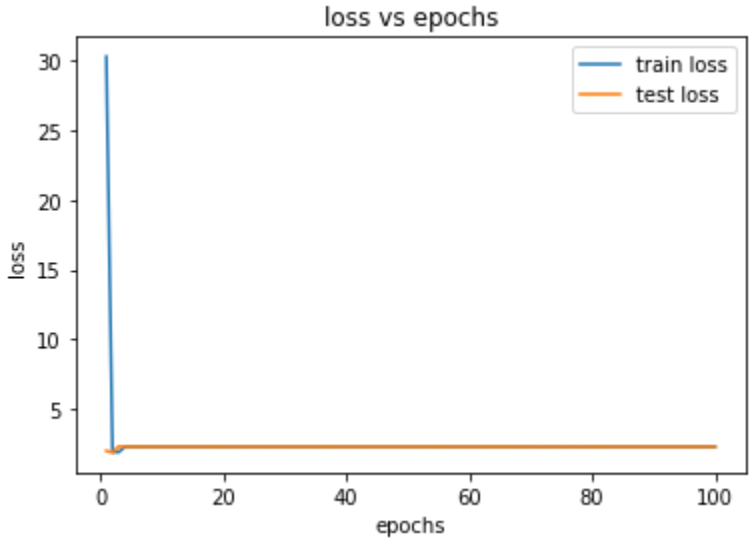
Sampling domain:

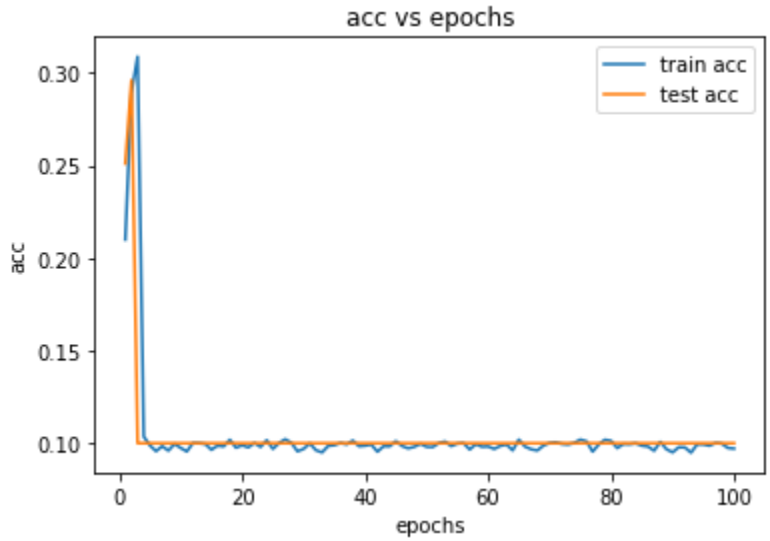
```
TRIALS = 10

HYPERPARAMS = {
    'lr': [0.1, 0.05, 0.01, 0.005, 0.001, 0.0001],
    'eps': [1e-7, 1e-8, 1e-9],
    'weight_decay': [1e-4, 1e-5, 1e-6]
}
```

➔ **RMSProp**

Default hyperparameters:

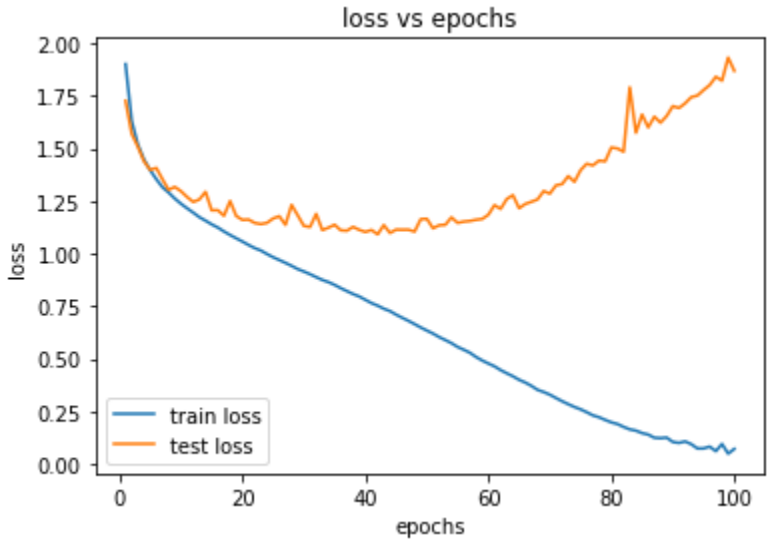
```
{lr: 0.01, alpha: 0.99, eps: 1e-8}
```

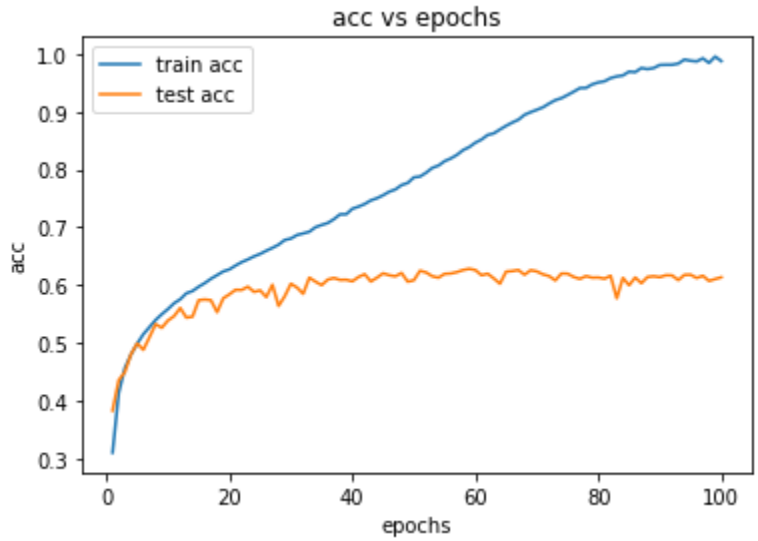




Optimal hyperparameters:

```
{alpha: 0.999, eps: 1e-08, lr: 0.0001, weight_decay: 1e-06}
```





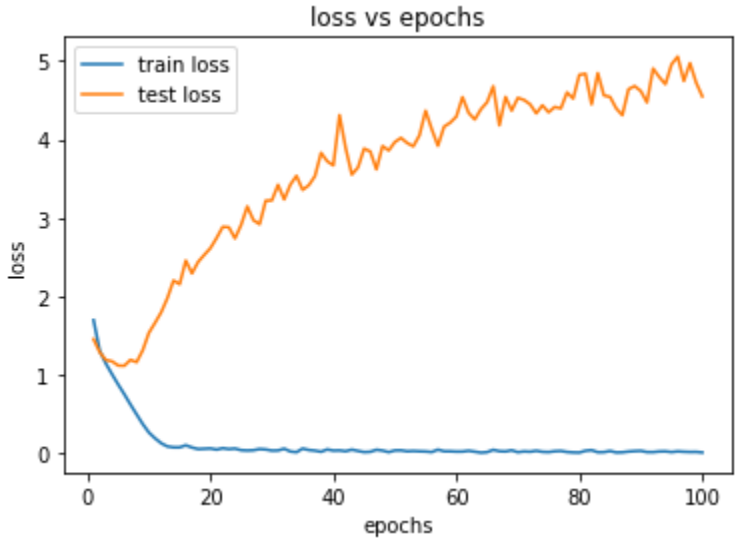
Sampling domain:

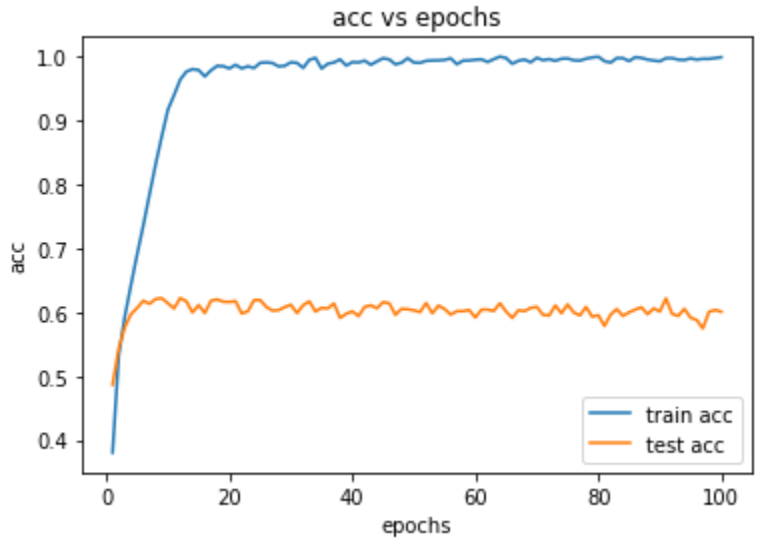
```
TRIALS = 10

HYPERPARAMS = {
    'lr': [0.1, 0.05, 0.01, 0.005, 0.001, 0.0001],
    'alpha': [0.9, 0.99, 0.999],
    'eps': [1e-7, 1e-8, 1e-9],
    'weight_decay': [1e-4, 1e-5, 1e-6]
}
```

➔ **ADAM**

Default hyperparameters:

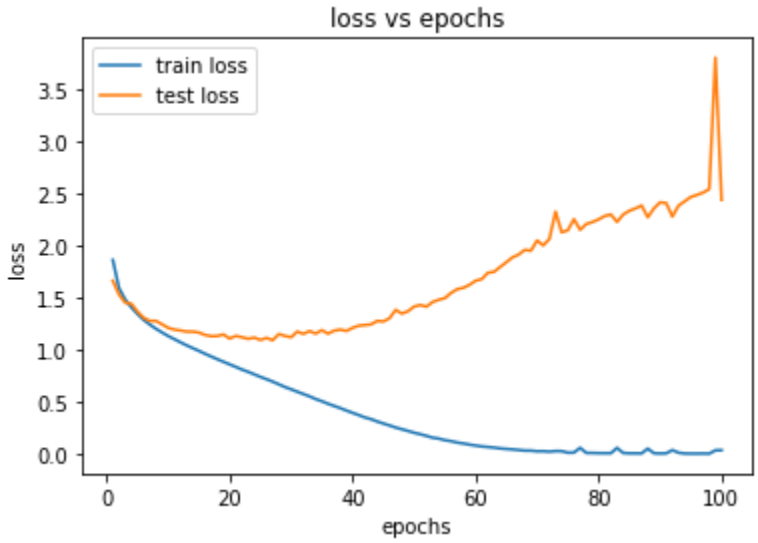
```
{lr: 0.001, betas: (0.9, 0.999), eps: 1e-8}
```

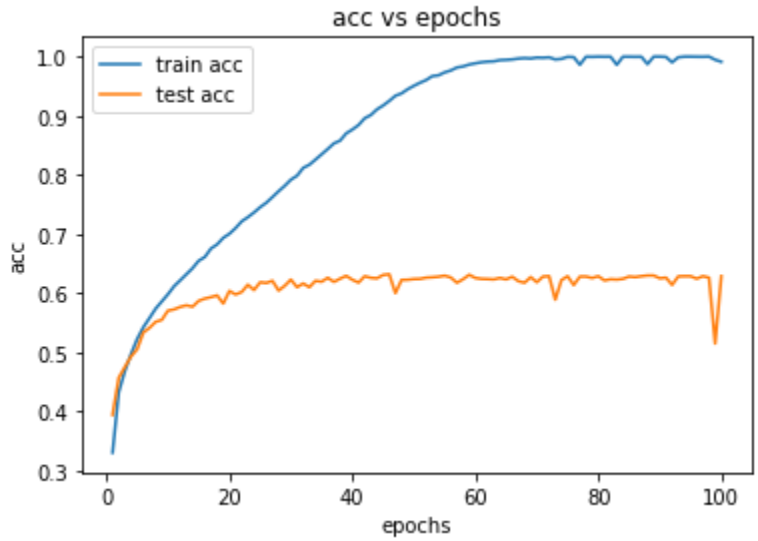




Optimal hyperparameters:

```
{beta1: 0.7, beta2: 0.999, eps: 1e-07, lr: 0.0001, weight_decay: 1e-06}
```





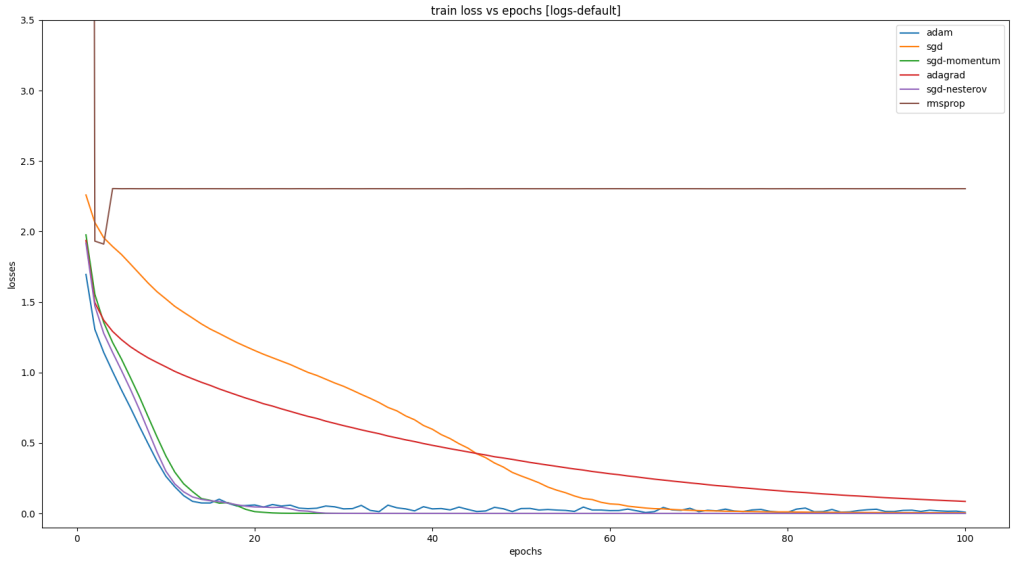
Sampling domain:

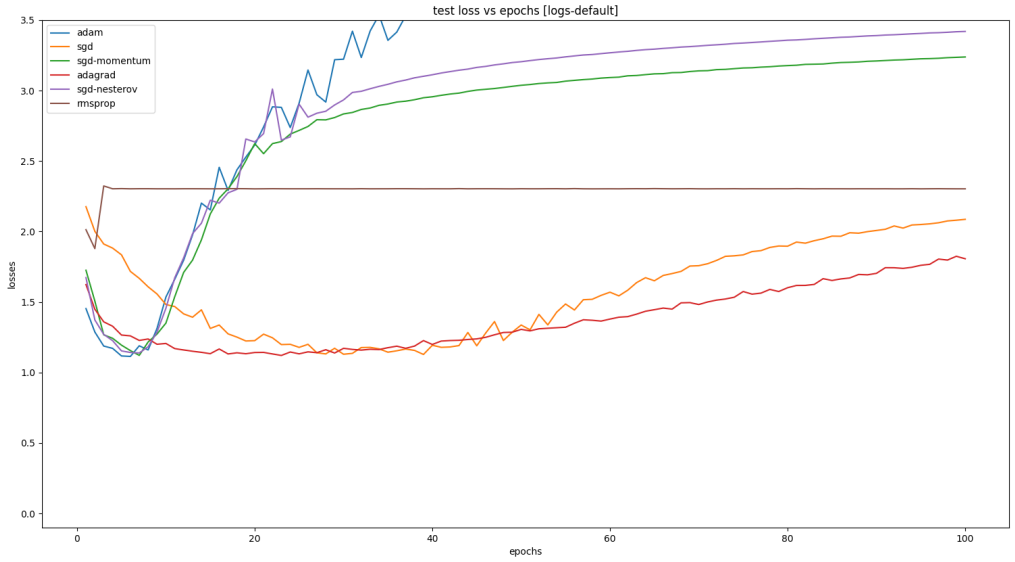
```
TRIALS = 10

HYPERPARAMS = {
    'lr': [0.01, 0.005, 0.001, 0.0001],
    'beta1': [0.5, 0.7, 0.9],
    'beta2': [0.9, 0.95, 0.999],
    'eps': [1e-7, 1e-8, 1e-9],
    'weight_decay': [1e-5, 1e-6]
}
```

Comparison between optimizers:

The training logs of different optimizers are parsed using *plot.py* for comparison.

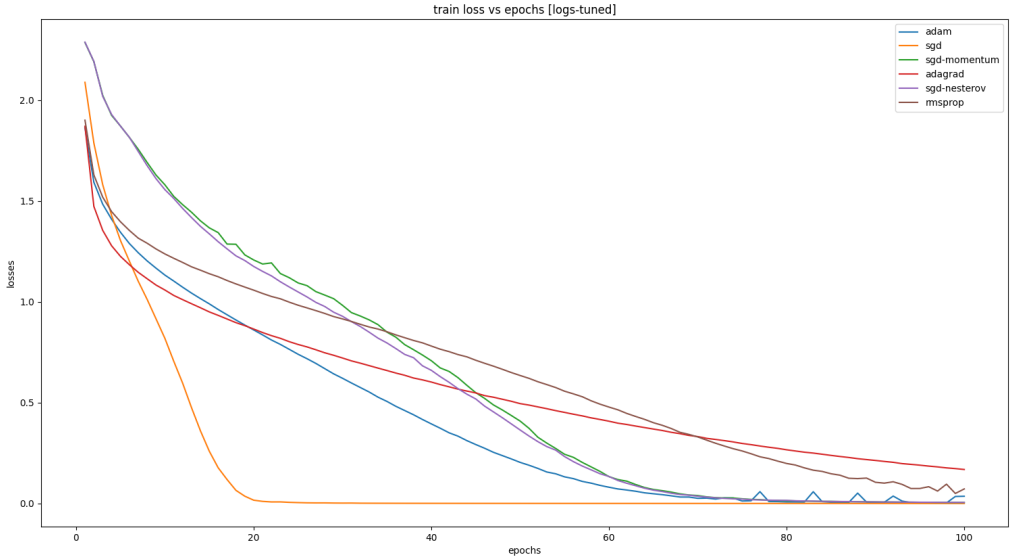
Using default hyperparameters:

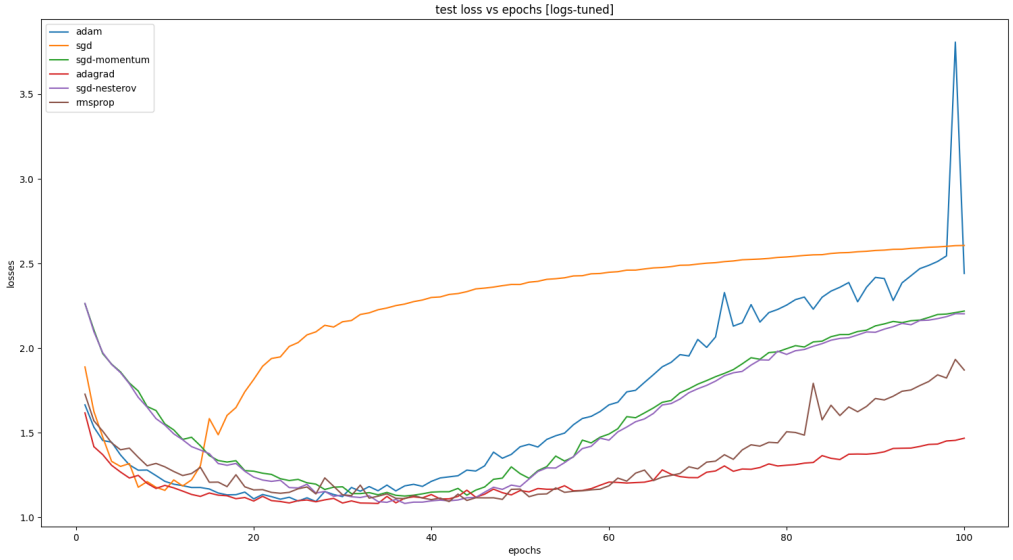




RMSprop effectively shows no convergence; it starts with a high loss, dips down, and then flatlines at high train error. ADAM, SGD-momentum & SGD-Nesterov have a higher rate of convergence compared to vanilla SGD & Adagrad. Although, in terms of generalizability, Adagrad performs the best, followed by vanilla SGD.

Using optimal hyperparameters:





After tuning, RMSprop showcased convergence. The most probable cause for this is the increase in the value of alpha in the optimal hyperparameters. Optimal hyperparameters were chosen based on their convergence and generalizability. Compared to no tuning, losses on the testing set are decreased and a smoother convergence trajectory is observed.