Assignment 4

In this assignment, we have worked on image classification with deep learning such as CNN and Transfer Learning. The models are implemented using pytorch lightning. Parts of code are adapted from the github on which the assignment was uploaded. The coding part is done on the google colab using T4 GPU as runtime.

The datasets used are Imagenette and CIFAR 10. These are imported from torchvision library.

CNN:

In this part, the dataset is transformed using Resize, tensor and normalize using transforms. 90% of data is training and 10% for validation. Batch size is 128 and num of workers is set to 8. Early stopping is configured with patience set to 5. The architecture for this network is built using Conv2d , ReLU and Maxpool2D as shown below.

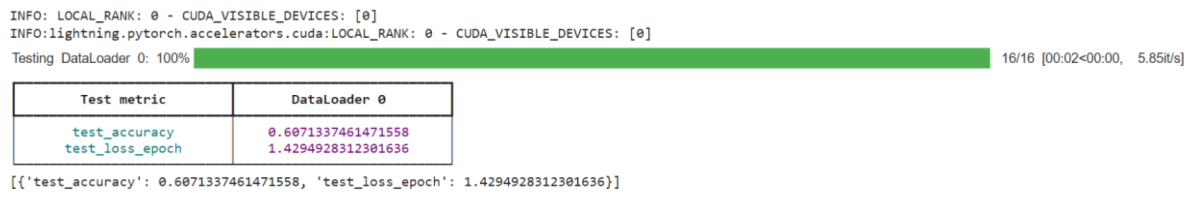
```
nn.Conv2d(3, 32, padding = 1, kernel_size = 3),
nn.ReLU(),
nn.MaxPool2d(2),
nn.Conv2d(32, 64, padding = 1, kernel_size = 3),
nn.ReLU(),
nn.MaxPool2d(2)
```

Output:

```
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used:
True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU  available:  False,  using:  0
TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU  available:  False,  using:  0
IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU  available:  False,  using:  0
HPUs
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK:           0             -
CUDA_VISIBLE_DEVICES: [0]
INFO:
  | Name     | Type                 | Params
-----------------------------------------------
0 | nw       | Sequential           | 19.4 K
1 | flat     | Flatten              | 0
2 | lay      | Sequential           | 803 K
```

```
3 | lay_con | Linear               | 2.6 K
4 | acc     | MulticlassAccuracy | 0
-----------------------------------------------
825 K     Trainable params
0         Non-trainable params
825 K     Total params
3.300     Total estimated model params size (MB)
INFO:lightning.pytorch.callbacks.model_summary:
  | Name    | Type                 | Params
-----------------------------------------------
0 | nw      | Sequential           | 19.4 K
1 | flat    | Flatten              | 0
2 | lay     | Sequential           | 803 K
3 | lay_con | Linear               | 2.6 K
4 | acc     | MulticlassAccuracy | 0
-----------------------------------------------
825 K     Trainable params
0         Non-trainable params
825 K     Total params
3.300     Total estimated model params size (MB)
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:558:
UserWarning: This DataLoader will create 8 worker processes in total. Our
suggested max number of worker in current system is 2, which is smaller than
what this DataLoader is going to create. Please be aware that excessive
worker creation might get DataLoader running slow or even freeze, lower the
worker number to avoid potential slowness/freeze if necessary.
  warnings.warn(_create_warning_msg(
/usr/lib/python3.10/multiprocessing/popen_fork.py:66:       RuntimeWarning:
os.fork() was called. os.fork() is incompatible with multithreaded code, and
JAX is multithreaded, so this will likely lead to a deadlock.
  self.pid = os.fork()
Epoch 9: 100%
 67/67 [00:08<00:00,  7.87it/s, v_num=0]
/usr/lib/python3.10/multiprocessing/popen_fork.py:66:       RuntimeWarning:
os.fork() was called. os.fork() is incompatible with multithreaded code, and
JAX is multithreaded, so this will likely lead to a deadlock.
  self.pid = os.fork()
{'train_loss': tensor(0.4154),
 'train_loss_step': tensor(0.4685),
 'val_accuracy': tensor(0.6156),
 'val_loss': tensor(1.3544),
 'val_loss_epoch': tensor(1.3544),
 'train_loss_epoch': tensor(0.4154)}
```

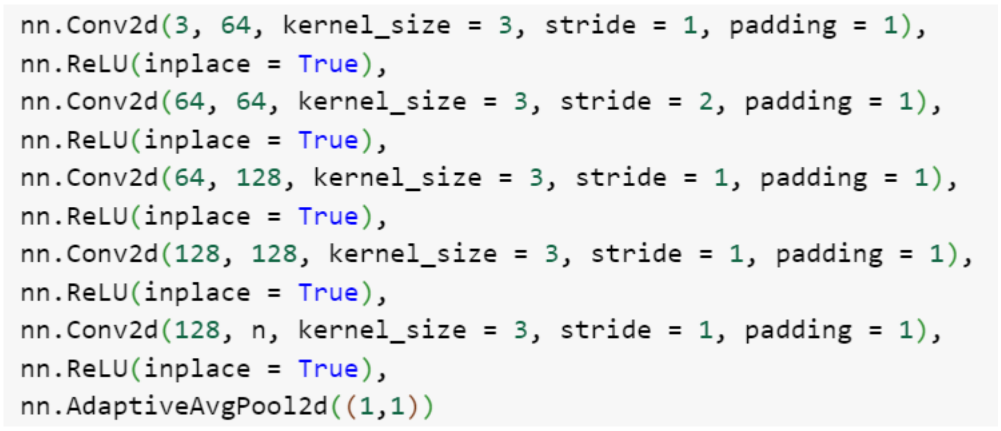
Test Results:



```
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Testing DataLoader 0: 100%                                16/16 [00:02<00:00,  5.85it/s]
┌──────────────────────┬──────────────────────┐
│     Test metric      │     DataLoader 0     │
├──────────────────────┼──────────────────────┤
│    test_accuracy     │  0.6071337461471558  │
│   test_loss_epoch    │  1.4294928312301636  │
└──────────────────────┴──────────────────────┘
[{'test_accuracy': 0.6071337461471558, 'test_loss_epoch': 1.4294928312301636}]
```

All Convolutional Net:

We have same transformed dataset for this network. The network architecture is as follows:



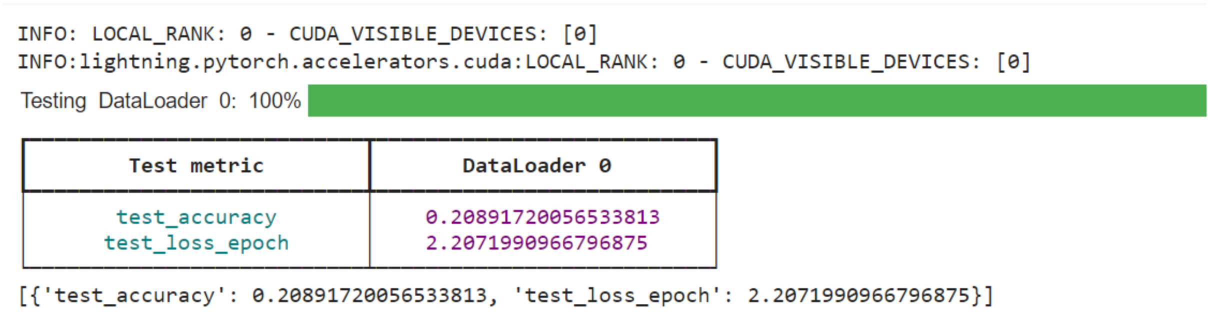
```
nn.Conv2d(3, 64, kernel_size = 3, stride = 1, padding = 1),
nn.ReLU(inplace = True),
nn.Conv2d(64, 64, kernel_size = 3, stride = 2, padding = 1),
nn.ReLU(inplace = True),
nn.Conv2d(64, 128, kernel_size = 3, stride = 1, padding = 1),
nn.ReLU(inplace = True),
nn.Conv2d(128, 128, kernel_size = 3, stride = 1, padding = 1),
nn.ReLU(inplace = True),
nn.Conv2d(128, n, kernel_size = 3, stride = 1, padding = 1),
nn.ReLU(inplace = True),
nn.AdaptiveAvgPool2d((1,1))
```

Training and validation output:

```
INFO: GPU available: True (cuda), used: True
INFO:lightning.pytorch.utilities.rank_zero:GPU available: True (cuda), used:
True
INFO: TPU available: False, using: 0 TPU cores
INFO:lightning.pytorch.utilities.rank_zero:TPU available: False, using: 0
TPU cores
INFO: IPU available: False, using: 0 IPUs
INFO:lightning.pytorch.utilities.rank_zero:IPU available: False, using: 0
IPUs
INFO: HPU available: False, using: 0 HPUs
INFO:lightning.pytorch.utilities.rank_zero:HPU available: False, using: 0
HPUs
/usr/local/lib/python3.10/dist-
packages/lightning/pytorch/loops/utilities.py:73: `max_epochs` was not set.
Setting it to 1000 epochs. To train without an epoch limit, set `max_epochs=-
1`.
/usr/local/lib/python3.10/dist-
packages/lightning/pytorch/callbacks/model_checkpoint.py:653: Checkpoint
directory /content/lightning_logs/version_0/checkpoints exists and is not
empty.
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 -
CUDA_VISIBLE_DEVICES: [0]
INFO:
  | Name | Type               | Params
--------------------------------------------
0 | feat | Sequential         | 271 K
1 | acc  | MulticlassAccuracy | 0
--------------------------------------------
271 K     Trainable params
0         Non-trainable params
271 K     Total params
1.087     Total estimated model params size (MB)
INFO:lightning.pytorch.callbacks.model_summary:
  | Name | Type               | Params
--------------------------------------------
```

```
0 | feat | Sequential           | 271 K
1 | acc  | MulticlassAccuracy | 0
--------------------------------------------
271 K     Trainable params
0         Non-trainable params
271 K     Total params
1.087     Total estimated model params size (MB)
 67/67 [00:09<00:00, 6.87it/s, v_num=1]
{'train_loss': tensor(2.2657),
 'train_loss_step': tensor(2.2465),
 'val_accuracy': tensor(0.2175),
 'val_loss': tensor(2.2126),
 'val_loss_epoch': tensor(2.2126),
 'train_loss_epoch': tensor(2.2657)}
```

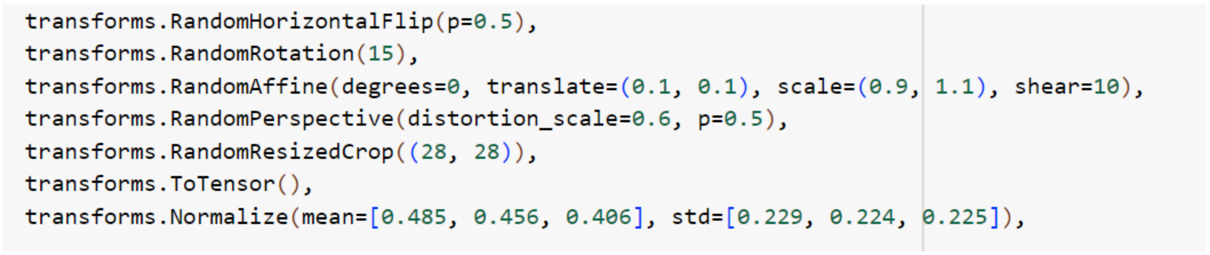
Test output:



```
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Testing DataLoader 0: 100%
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
┃        Test metric        ┃       DataLoader 0        ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
│       test_accuracy       │    0.20891720056533813    │
│      test_loss_epoch      │    2.2071990966796875     │
└───────────────────────────┴───────────────────────────┘
[{'test_accuracy': 0.20891720056533813, 'test_loss_epoch': 2.2071990966796875}]
```

Regularization:

For this part, I have used the basic cnn as the model. The regularization is added in the form of data augmentation as shown below.



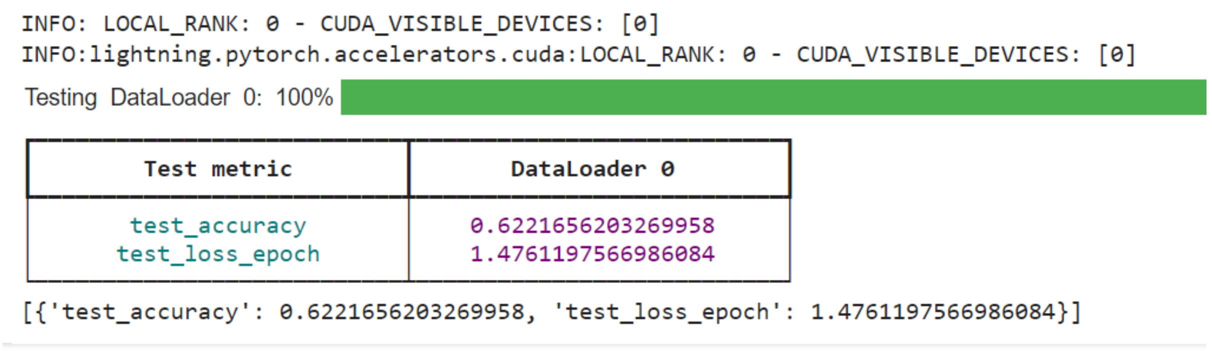
```
transforms.RandomHorizontalFlip(p=0.5),
transforms.RandomRotation(15),
transforms.RandomAffine(degrees=0, translate=(0.1, 0.1), scale=(0.9, 1.1), shear=10),
transforms.RandomPerspective(distortion_scale=0.6, p=0.5),
transforms.RandomResizedCrop((28, 28)),
transforms.ToTensor(),
transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
```

Training output:

```
INFO:
  | Name     | Type                   | Params
-------------------------------------------------
0 | nw       | Sequential             | 19.4 K
1 | flat     | Flatten                | 0
2 | lay      | Sequential             | 803 K
3 | lay_con | Linear             | 2.6 K
4 | acc      | MulticlassAccuracy | 0
----------------------------------------------
```

```
825 K     Trainable params
0         Non-trainable params
825 K     Total params
3.300     Total estimated model params size (MB)
INFO:lightning.pytorch.callbacks.model_summary:
  | Name    | Type               | Params
-----------------------------------------------
0 | nw      | Sequential         | 19.4 K
1 | flat    | Flatten            | 0
2 | lay     | Sequential         | 803 K
3 | lay_con | Linear             | 2.6 K
4 | acc     | MulticlassAccuracy | 0
-----------------------------------------------
825 K     Trainable params
0         Non-trainable params
825 K     Total params
3.300     Total estimated model params size (MB)
Epoch 10: 100%
 67/67 [00:09<00:00, 7.04it/s, v_num=2]
{'train_loss': tensor(0.2761),
 'train_loss_step': tensor(0.4198),
 'val_accuracy': tensor(0.6399),
 'val_loss': tensor(1.4153),
 'val_loss_epoch': tensor(1.4153),
 'train_loss_epoch': tensor(0.2761)}
```

Test Output:

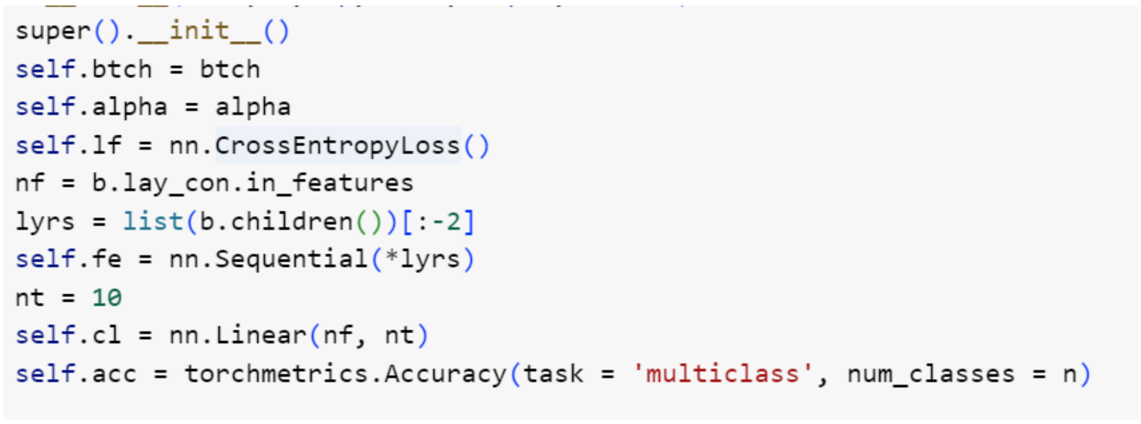


```
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Testing DataLoader 0: 100%

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
┃        Test metric        ┃       DataLoader 0        ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
│       test_accuracy       │    0.6221656203269958     │
│      test_loss_epoch      │    1.4761197566986084     │
└───────────────────────────┴───────────────────────────┘
[{'test_accuracy': 0.6221656203269958, 'test_loss_epoch': 1.4761197566986084}]
```

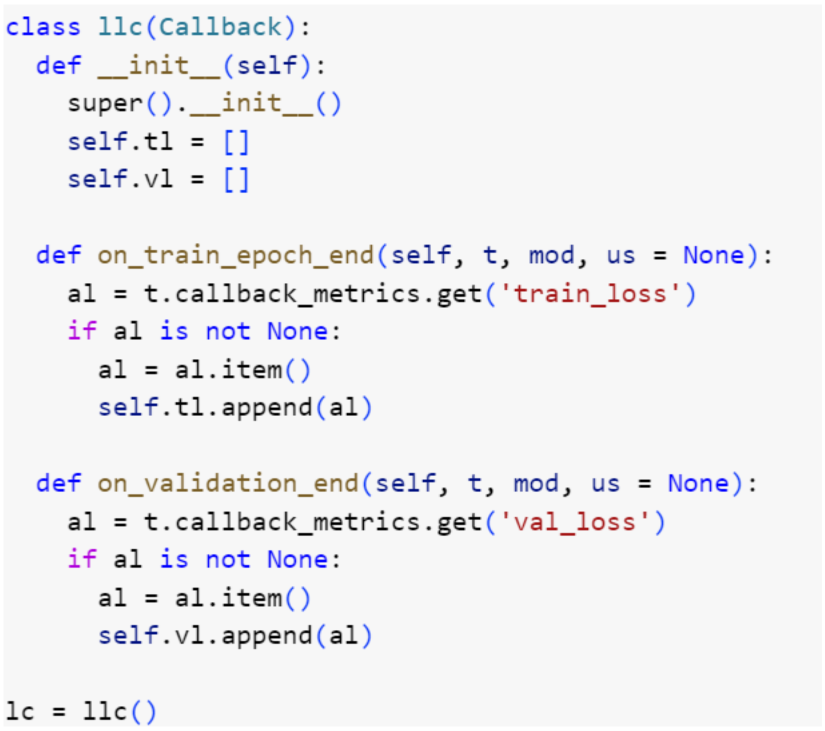
Transfer Learning:

In this section, We have used CNN model as our pre trained model for the transfer learning model. The dataset is augmented in the same way as above for this dataset. The architecture for this network is shown below:



```
super().__init__()
self.btch = btch
self.alpha = alpha
self.lf = nn.CrossEntropyLoss()
nf = b.lay_con.in_features
lyrs = list(b.children())[:-2]
self.fe = nn.Sequential(*lyrs)
nt = 10
self.cl = nn.Linear(nf, nt)
self.acc = torchmetrics.Accuracy(task = 'multiclass', num_classes = n)
```

The LossLogger call back is implemented as shown below:



```
class llc(Callback):
  def __init__(self):
    super().__init__()
    self.tl = []
    self.vl = []

  def on_train_epoch_end(self, t, mod, us = None):
    al = t.callback_metrics.get('train_loss')
    if al is not None:
      al = al.item()
      self.tl.append(al)

  def on_validation_end(self, t, mod, us = None):
    al = t.callback_metrics.get('val_loss')
    if al is not None:
      al = al.item()
      self.vl.append(al)

lc = llc()
```

Training output:

```
INFO:lightning.pytorch.callbacks.model_summary:
  | Name | Type               | Params
--------------------------------------------
0 | lf   | CrossEntropyLoss   | 0
1 | fe   | Sequential         | 822 K
2 | cl   | Linear             | 2.6 K
3 | acc  | MulticlassAccuracy | 0
--------------------------------------------
```

```
825 K     Trainable params
0         Non-trainable params
825 K     Total params
3.300     Total estimated model params size (MB)
```

Epoch 30: 100%
 704/704 [00:21<00:00, 32.71it/s, v_num=3, train_loss_step=2.030, val_loss_step=1.270, val_loss_epoch=1.330, train_loss_epoch=1.310]

```
{'train_loss': tensor(1.3101),
 'train_loss_step': tensor(2.0292),
 'val_loss': tensor(1.3314),
 'val_loss_epoch': tensor(1.3314),
 'train_loss_epoch': tensor(1.3101)}
```

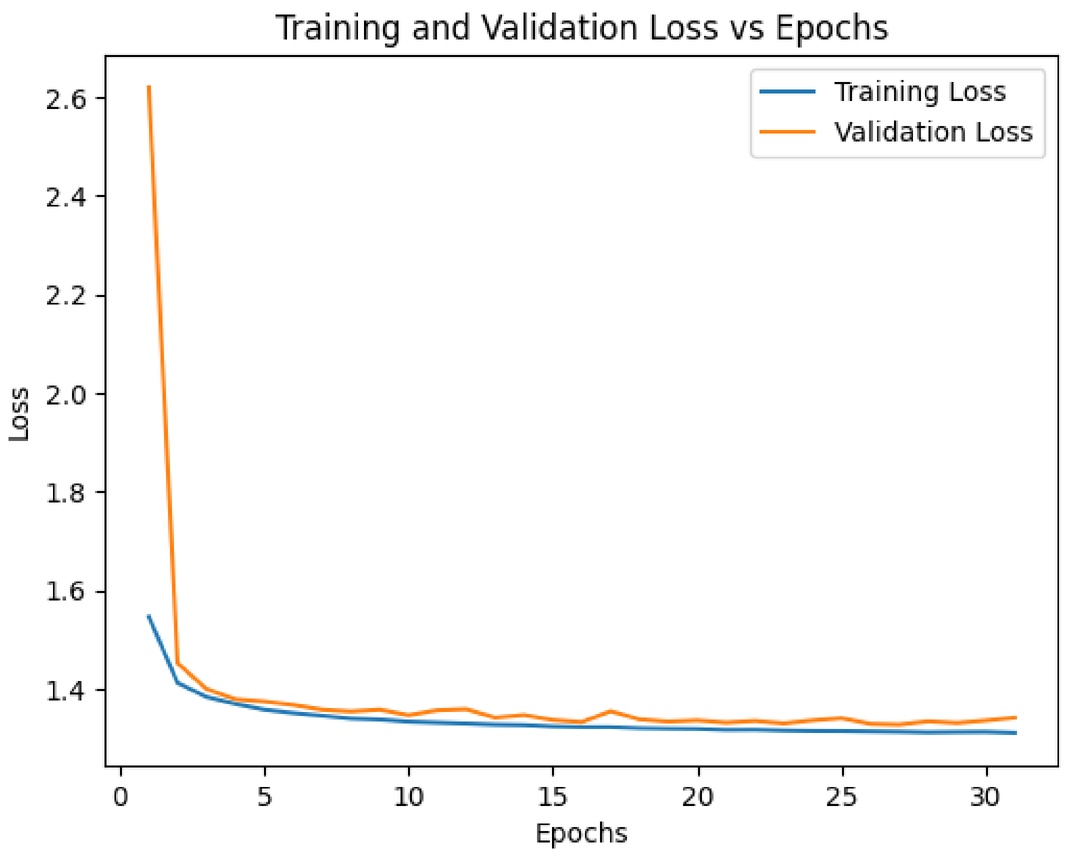
Test Output:

```
INFO: LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
INFO:lightning.pytorch.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Testing DataLoader 0: 100%
```

Test metric	DataLoader 0
test_accuracy	0.5234000086784363
test_loss	1.329429030418396

```
{'test_accuracy': tensor(0.5234), 'test_loss': tensor(1.3294)}
```

Training and Validation loss Vs Epochs plot:



From this plot we can say that the training and Validation loss are close to each other indicating that the model is generalising well on unseen data. There are no signs of potential overfitting or instability.

Conclusion:

In this assignment, we have worked on the Imagenette and CIFAR10 datasets by using deep learning models such as CNN and All Convolutional Net. We have used two callbacks: early stopping and checkpoint. We have transformed the data using transforms. For accuracy we have used torchmetrics. In the transfer learning section, we have used loss logger callback. We have achieved good results from the plot.