In [22]:
```
%load_ext autoreload
%autoreload 2
%matplotlib inline
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

In [23]:
```
import os

import torch
import torch.nn as nn
from dataloader.dataloader import load_data
import numpy as np

from util import train_validate
import matplotlib.pyplot as plt

torch.backends.cudnn.benchmark = True
```

Task 1

In [5]:
```
config = {
        'model' : 'basic',
        'dataset' : 'mnist',
        'batch_size' : 256,
        'num_workers' : 4,
        'lr' : 1e-1,
        'epochs' : 30,
        'criterion' : nn.NLLLoss,
        'optimizer' : 'SGD',
        'device' : 'cuda' if torch.cuda.is_available() else 'cpu',
        'input_channels' : 1,
        'num_classes' : 10,
        'pin_memory' : True if torch.cuda.is_available() else False,
        'lr_scheduler' : "ReduceLROnPlateau"
    }

val_loss_array, val_accuracy_array, train_loss_array, train_accuracy_array, flops, params, inference_time = train_valid
```

```
Train loss: 0.0374, Train accuracy: 98.79
Validation loss: 0.0288, Validation accuracy: 99.04
-------------------------------------------------
[INFO]: Epoch 13/30
Train loss: 0.0361, Train accuracy: 98.87
Validation loss: 0.0372, Validation accuracy: 98.74
-------------------------------------------------
[INFO]: Epoch 14/30
Train loss: 0.0343, Train accuracy: 98.89
Validation loss: 0.0302, Validation accuracy: 99.02
-------------------------------------------------
Epoch 00014: reducing learning rate of group 0 to 5.0000e-03.
[INFO]: Epoch 15/30
Train loss: 0.0290, Train accuracy: 99.06
Validation loss: 0.0264, Validation accuracy: 99.13
-------------------------------------------------
[INFO]: Epoch 16/30
Train loss: 0.0271, Train accuracy: 99.14
Validation loss: 0.0262, Validation accuracy: 99.09
-------------------------------------------------
[INFO]: Epoch 17/30
Train loss: 0.0266, Train accuracy: 99.19
Validation loss: 0.0262, Validation accuracy: 99.08
-------------------------------------------------
[INFO]: Epoch 18/30
Train loss: 0.0244, Train accuracy: 99.23
Validation loss: 0.0265, Validation accuracy: 99.07
-------------------------------------------------
[INFO]: Epoch 19/30
Train loss: 0.0250, Train accuracy: 99.20
Validation loss: 0.0264, Validation accuracy: 99.06
-------------------------------------------------
[INFO]: Epoch 20/30
Train loss: 0.0243, Train accuracy: 99.24
Validation loss: 0.0264, Validation accuracy: 99.07
-------------------------------------------------
Epoch 00020: reducing learning rate of group 0 to 2.5000e-04.
[INFO]: Epoch 21/30
Train loss: 0.0239, Train accuracy: 99.26
Validation loss: 0.0264, Validation accuracy: 99.07
-------------------------------------------------
[INFO]: Epoch 22/30
Train loss: 0.0232, Train accuracy: 99.25
Validation loss: 0.0264, Validation accuracy: 99.07
-------------------------------------------------
```

```
[INFO]: Epoch 23/30
Train loss: 0.0241, Train accuracy: 99.21
Validation loss: 0.0264, Validation accuracy: 99.07
-------------------------------------------------
[INFO]: Epoch 24/30
Train loss: 0.0242, Train accuracy: 99.25
Validation loss: 0.0264, Validation accuracy: 99.06
-------------------------------------------------
[INFO]: Epoch 25/30
Train loss: 0.0237, Train accuracy: 99.24
Validation loss: 0.0264, Validation accuracy: 99.06
-------------------------------------------------
Epoch 00025: reducing learning rate of group 0 to 1.2500e-05.
[INFO]: Epoch 26/30
Train loss: 0.0230, Train accuracy: 99.29
Validation loss: 0.0264, Validation accuracy: 99.06
-------------------------------------------------
[INFO]: Epoch 27/30
Train loss: 0.0233, Train accuracy: 99.26
Validation loss: 0.0264, Validation accuracy: 99.06
-------------------------------------------------
[INFO]: Epoch 28/30
Train loss: 0.0248, Train accuracy: 99.22
Validation loss: 0.0264, Validation accuracy: 99.06
-------------------------------------------------
[INFO]: Epoch 29/30
Train loss: 0.0233, Train accuracy: 99.29
Validation loss: 0.0264, Validation accuracy: 99.06
-------------------------------------------------
[INFO]: Epoch 30/30
Train loss: 0.0226, Train accuracy: 99.30
Validation loss: 0.0264, Validation accuracy: 99.06
-------------------------------------------------
Epoch 00030: reducing learning rate of group 0 to 6.2500e-07.
```

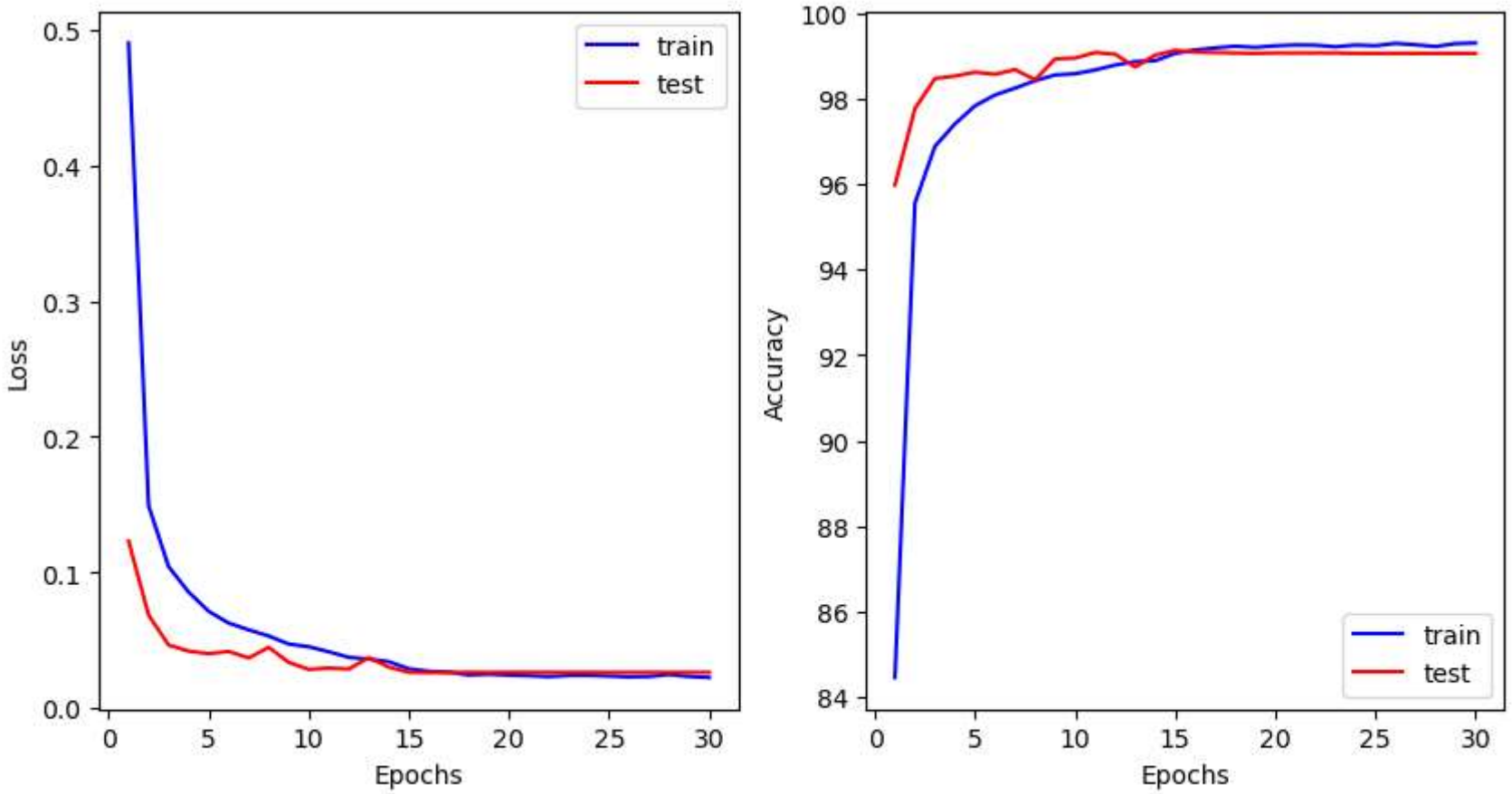
Loss vs Epochs

In [6]:

```
plt.figure(figsize=(10,5))

plt.subplot(121)
plt.plot(np.arange(len(train_loss_array)) + 1,train_loss_array,c='b',label='train')
plt.plot(np.arange(len(val_loss_array)) + 1,val_loss_array, c='r',label='test')
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

```
plt.legend()

plt.subplot(122)
plt.plot(np.arange(len(train_accuracy_array)) + 1,train_accuracy_array,c='b',label='train')
plt.plot(np.arange(len(val_accuracy_array)) + 1,val_accuracy_array, c='r',label='test')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```

Out[6]:
```
<matplotlib.legend.Legend at 0x1e10b69d710>
```



In [7]:
```
print(f"The Accuracy of the above model on test data is : {max(val_accuracy_array)}")
```

```
The Accuracy of the above model on test data is : 99.13
```

Initial Accuracy on MNIST: 99.13%

Task 2

MNIST

Change in Network

In [8]:
```
config = {
        'model' : 'resnet18',
        'dataset' : 'mnist',
        'batch_size' : 256,
        'num_workers' : 4,
        'lr' : 1e-1,
        'epochs' : 30,
        'criterion' : nn.NLLLoss,
        'optimizer' : 'SGD',
        'device' : 'cuda' if torch.cuda.is_available() else 'cpu',
        'input_channels' : 1,
        'num_classes' : 10,
        'pin_memory' : True if torch.cuda.is_available() else False,
        'lr_scheduler' : "ReduceLROnPlateau"
    }

val_loss_array, val_accuracy_array, train_loss_array, train_accuracy_array, flops, params, inference_time = train_valid
```

```
[INFO]: Epoch 1/30
Train loss: 0.3303, Train accuracy: 90.79
Validation loss: 0.0731, Validation accuracy: 97.94
------------------------------------------------
[INFO]: Epoch 2/30
Train loss: 0.0754, Train accuracy: 97.68
Validation loss: 0.0340, Validation accuracy: 98.84
------------------------------------------------
[INFO]: Epoch 3/30
Train loss: 0.0571, Train accuracy: 98.28
Validation loss: 0.1880, Validation accuracy: 94.54
------------------------------------------------
[INFO]: Epoch 4/30
Train loss: 0.0475, Train accuracy: 98.55
Validation loss: 0.0353, Validation accuracy: 98.88
------------------------------------------------
[INFO]: Epoch 5/30
Train loss: 0.0393, Train accuracy: 98.81
Validation loss: 0.0261, Validation accuracy: 99.24
------------------------------------------------
[INFO]: Epoch 6/30
Train loss: 0.0345, Train accuracy: 98.98
Validation loss: 0.0239, Validation accuracy: 99.22
------------------------------------------------
[INFO]: Epoch 7/30
Train loss: 0.0310, Train accuracy: 99.02
Validation loss: 0.1083, Validation accuracy: 97.10
------------------------------------------------
[INFO]: Epoch 8/30
Train loss: 0.0286, Train accuracy: 99.14
Validation loss: 0.0285, Validation accuracy: 99.09
------------------------------------------------
[INFO]: Epoch 9/30
Train loss: 0.0246, Train accuracy: 99.18
Validation loss: 0.0309, Validation accuracy: 99.06
------------------------------------------------
[INFO]: Epoch 10/30
Train loss: 0.0243, Train accuracy: 99.25
Validation loss: 0.0232, Validation accuracy: 99.28
------------------------------------------------
[INFO]: Epoch 11/30
Train loss: 0.0207, Train accuracy: 99.37
Validation loss: 0.0256, Validation accuracy: 99.15
------------------------------------------------
[INFO]: Epoch 12/30
```

```
Train loss: 0.0199, Train accuracy: 99.38
Validation loss: 0.0202, Validation accuracy: 99.39
-------------------------------------------------
[INFO]: Epoch 13/30
Train loss: 0.0195, Train accuracy: 99.38
Validation loss: 0.0273, Validation accuracy: 99.13
-------------------------------------------------
[INFO]: Epoch 14/30
Train loss: 0.0184, Train accuracy: 99.40
Validation loss: 0.0265, Validation accuracy: 99.28
-------------------------------------------------
[INFO]: Epoch 15/30
Train loss: 0.0163, Train accuracy: 99.46
Validation loss: 0.1415, Validation accuracy: 96.48
-------------------------------------------------
[INFO]: Epoch 16/30
Train loss: 0.0155, Train accuracy: 99.47
Validation loss: 0.0210, Validation accuracy: 99.41
-------------------------------------------------
Epoch 00016: reducing learning rate of group 0 to 5.0000e-03.
[INFO]: Epoch 17/30
Train loss: 0.0120, Train accuracy: 99.59
Validation loss: 0.0194, Validation accuracy: 99.46
-------------------------------------------------
[INFO]: Epoch 18/30
Train loss: 0.0101, Train accuracy: 99.68
Validation loss: 0.0193, Validation accuracy: 99.45
-------------------------------------------------
[INFO]: Epoch 19/30
Train loss: 0.0104, Train accuracy: 99.67
Validation loss: 0.0185, Validation accuracy: 99.47
-------------------------------------------------
[INFO]: Epoch 20/30
Train loss: 0.0092, Train accuracy: 99.71
Validation loss: 0.0190, Validation accuracy: 99.51
-------------------------------------------------
[INFO]: Epoch 21/30
Train loss: 0.0096, Train accuracy: 99.68
Validation loss: 0.0183, Validation accuracy: 99.53
-------------------------------------------------
[INFO]: Epoch 22/30
Train loss: 0.0088, Train accuracy: 99.72
Validation loss: 0.0186, Validation accuracy: 99.50
-------------------------------------------------
[INFO]: Epoch 23/30
```

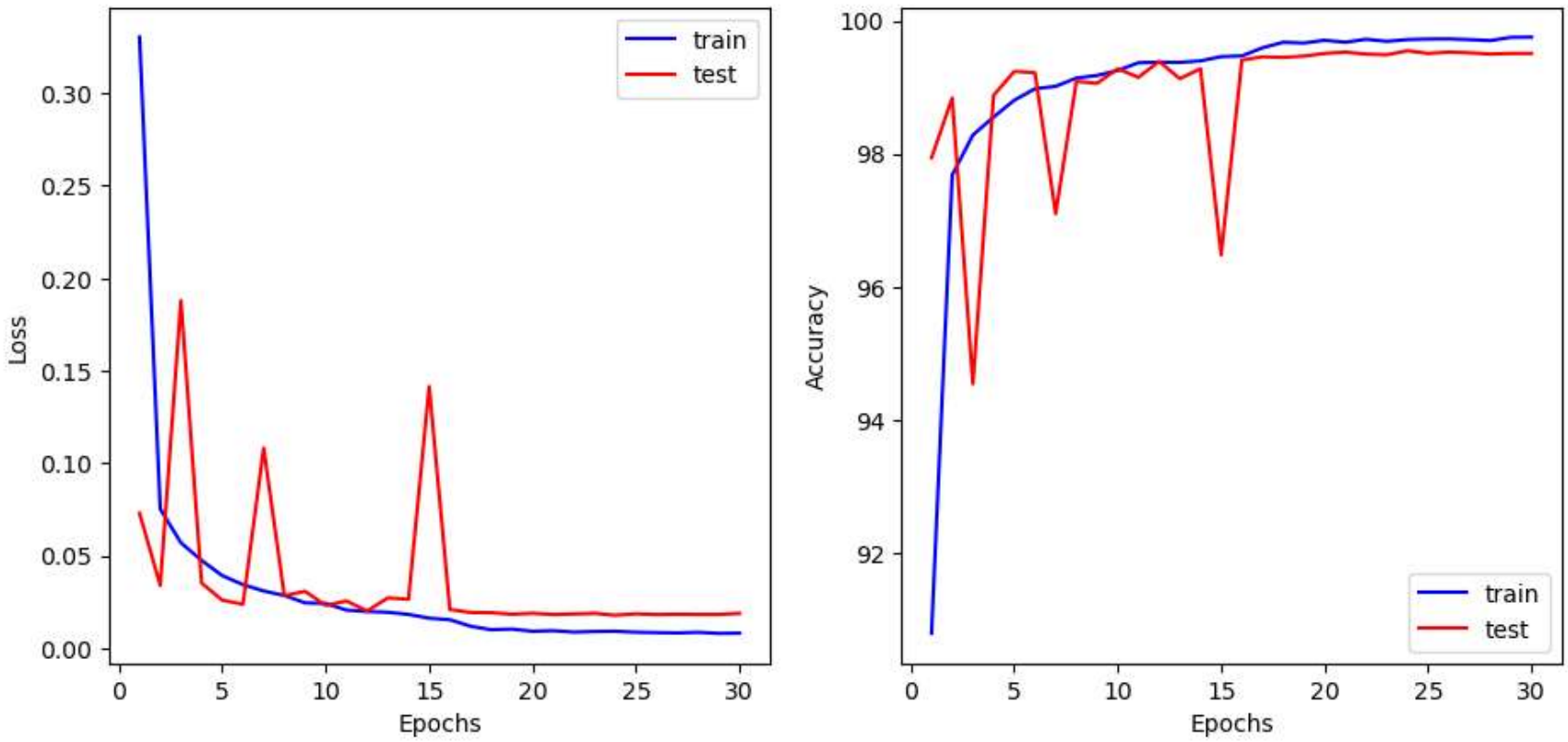
```
Train loss: 0.0091, Train accuracy: 99.69
Validation loss: 0.0189, Validation accuracy: 99.49
-------------------------------------------------
[INFO]: Epoch 24/30
Train loss: 0.0093, Train accuracy: 99.72
Validation loss: 0.0179, Validation accuracy: 99.55
-------------------------------------------------
[INFO]: Epoch 25/30
Train loss: 0.0087, Train accuracy: 99.73
Validation loss: 0.0187, Validation accuracy: 99.51
-------------------------------------------------
[INFO]: Epoch 26/30
Train loss: 0.0085, Train accuracy: 99.73
Validation loss: 0.0183, Validation accuracy: 99.53
-------------------------------------------------
[INFO]: Epoch 27/30
Train loss: 0.0083, Train accuracy: 99.72
Validation loss: 0.0185, Validation accuracy: 99.52
-------------------------------------------------
[INFO]: Epoch 28/30
Train loss: 0.0087, Train accuracy: 99.70
Validation loss: 0.0184, Validation accuracy: 99.50
-------------------------------------------------
Epoch 00028: reducing learning rate of group 0 to 2.5000e-04.
[INFO]: Epoch 29/30
Train loss: 0.0081, Train accuracy: 99.75
Validation loss: 0.0184, Validation accuracy: 99.51
-------------------------------------------------
[INFO]: Epoch 30/30
Train loss: 0.0082, Train accuracy: 99.75
Validation loss: 0.0190, Validation accuracy: 99.51
-------------------------------------------------
```

In [9]:

```python
plt.figure(figsize=(11,5))

plt.subplot(121)
plt.plot(np.arange(len(train_loss_array)) + 1,train_loss_array,c='b',label='train')
plt.plot(np.arange(len(train_loss_array)) + 1,val_loss_array, c='r',label='test')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(122)
plt.plot(np.arange(len(train_loss_array)) + 1,train_accuracy_array,c='b',label='train')
plt.plot(np.arange(len(train_loss_array)) + 1,val_accuracy_array, c='r',label='test')
```

```
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```

Out[9]:
```
<matplotlib.legend.Legend at 0x1e1141bbf50>
```



In [10]:
```
print(f"The Accuracy of the above model on test data is : {max(val_accuracy_array)}")
```

```
The Accuracy of the above model on test data is : 99.55
```

Change in Optimizer

In [25]:
```
config = {
        'model' : 'basic',
        'dataset' : 'mnist',
        'batch_size' : 256,
        'num_workers' : 4,
        'lr' : 1e-3,
```

```
        'epochs' : 50,
        'criterion' : nn.NLLLoss,
        'optimizer' : 'SGD',
        'device' : 'cuda' if torch.cuda.is_available() else 'cpu',
        'input_channels' : 1,
        'num_classes' : 10,
        'pin_memory' : True if torch.cuda.is_available() else False,
        'lr_scheduler' : "ReduceLROnPlateau"
    }

optimizers = ['Adam', 'SGD', 'RMSprop',]

val_loss_arrays, val_accuracy_arrays, train_loss_arrays, train_accuracy_arrays = [], [], [], []

for optimizer in optimizers:
    config['optimizer'] = optimizer
    val_loss_array, val_accuracy_array, train_loss_array, train_accuracy_array, flops, params, inference_time = train_v

    val_loss_arrays.append(val_loss_array)
    val_accuracy_arrays.append(val_accuracy_array)
```

```
Train loss: 0.0438, Train accuracy: 98.66
Validation loss: 0.0319, Validation accuracy: 98.92
-------------------------------------------------
[INFO]: Epoch 8/50
Train loss: 0.0403, Train accuracy: 98.74
Validation loss: 0.0297, Validation accuracy: 99.14
-------------------------------------------------
[INFO]: Epoch 9/50
Train loss: 0.0382, Train accuracy: 98.81
Validation loss: 0.0302, Validation accuracy: 99.10
-------------------------------------------------
[INFO]: Epoch 10/50
Train loss: 0.0336, Train accuracy: 98.93
Validation loss: 0.0287, Validation accuracy: 99.11
-------------------------------------------------
[INFO]: Epoch 11/50
Train loss: 0.0296, Train accuracy: 99.04
Validation loss: 0.0299, Validation accuracy: 99.12
-------------------------------------------------
[INFO]: Epoch 12/50
Train loss: 0.0280, Train accuracy: 99.10
Validation loss: 0.0282, Validation accuracy: 99.17
-------------------------------------------------
[INFO]: Epoch 13/50
Train loss: 0.0277, Train accuracy: 99.11
Validation loss: 0.0299, Validation accuracy: 99.14
-------------------------------------------------
[INFO]: Epoch 14/50
Train loss: 0.0250, Train accuracy: 99.23
Validation loss: 0.0277, Validation accuracy: 99.17
-------------------------------------------------
[INFO]: Epoch 15/50
Train loss: 0.0253, Train accuracy: 99.17
Validation loss: 0.0303, Validation accuracy: 99.16
-------------------------------------------------
[INFO]: Epoch 16/50
Train loss: 0.0231, Train accuracy: 99.27
Validation loss: 0.0309, Validation accuracy: 99.12
-------------------------------------------------
[INFO]: Epoch 17/50
Train loss: 0.0211, Train accuracy: 99.34
Validation loss: 0.0321, Validation accuracy: 99.19
-------------------------------------------------
[INFO]: Epoch 18/50
Train loss: 0.0205, Train accuracy: 99.36
```

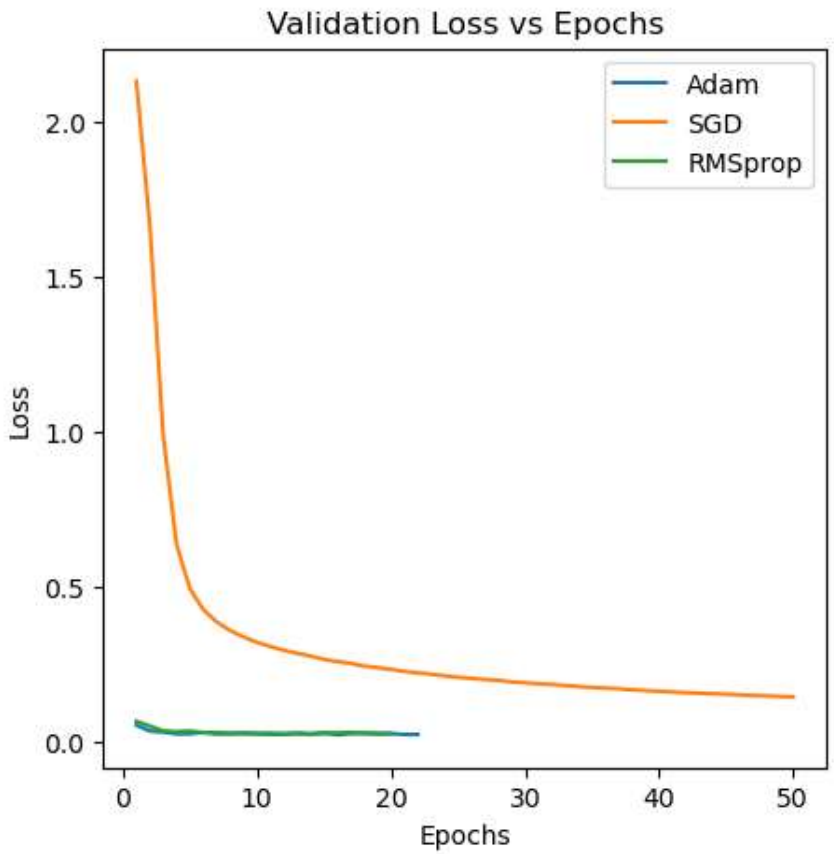
```
Validation loss: 0.0293, Validation accuracy: 99.25
-------------------------------------------------
Epoch 00018: reducing learning rate of group 0 to 5.0000e-05.
[INFO]: Epoch 19/50
Train loss: 0.0159, Train accuracy: 99.49
Validation loss: 0.0292, Validation accuracy: 99.24
-------------------------------------------------
[INFO]: Epoch 20/50
Train loss: 0.0143, Train accuracy: 99.49
Validation loss: 0.0297, Validation accuracy: 99.23
-------------------------------------------------
[INFO]: Early stopping
```

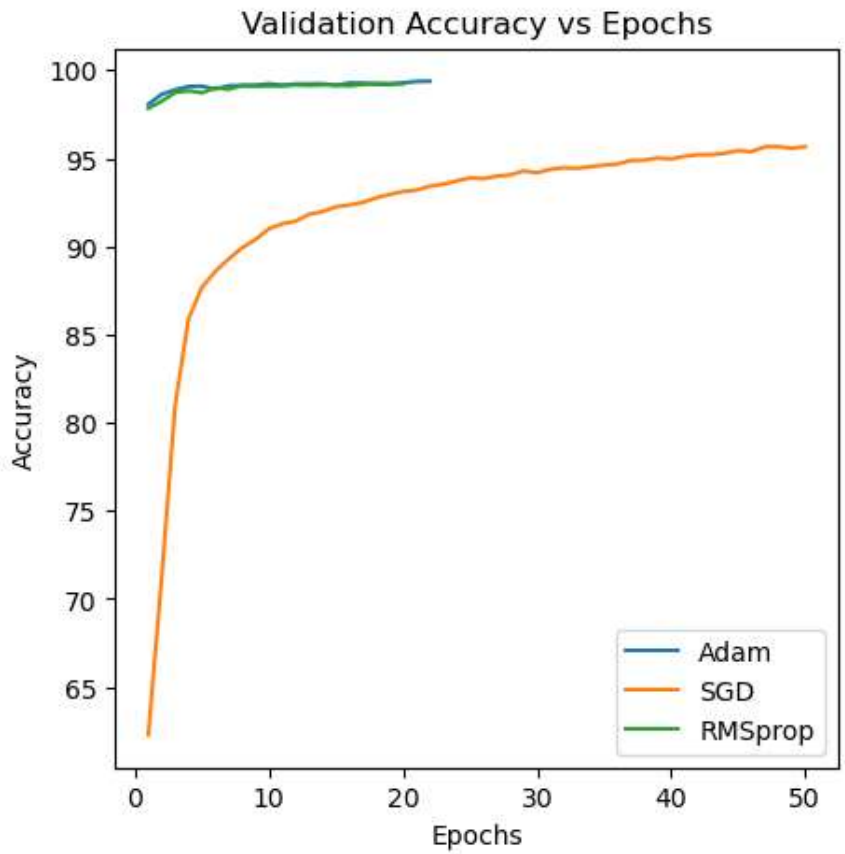
In [26]:

```python
plt.figure(figsize=(11,5))

plt.subplot(121)
for i in range(len(optimizers)):
    plt.plot(np.arange(len(val_loss_arrays[i])) + 1,val_loss_arrays[i], label=f"{optimizers[i]}")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title("Validation Loss vs Epochs")
plt.legend()

plt.subplot(122)
for i in range(len(optimizers)):
    plt.plot(np.arange(len(val_accuracy_arrays[i])) + 1,val_accuracy_arrays[i], label=f"{optimizers[i]}")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title("Validation Accuracy vs Epochs")
plt.legend()
```

Out[26]:

```
<matplotlib.legend.Legend at 0x1e1234c5950>
```





In [27]:
```
for i in range(len(optimizers)):
    print(f"Accuracy of above model using {optimizers[i]} : {max(val_accuracy_arrays[i])}")
```

```
Accuracy of above model using Adam : 99.38
Accuracy of above model using SGD : 95.67
Accuracy of above model using RMSprop : 99.25
```

Change In Learning Scheduler

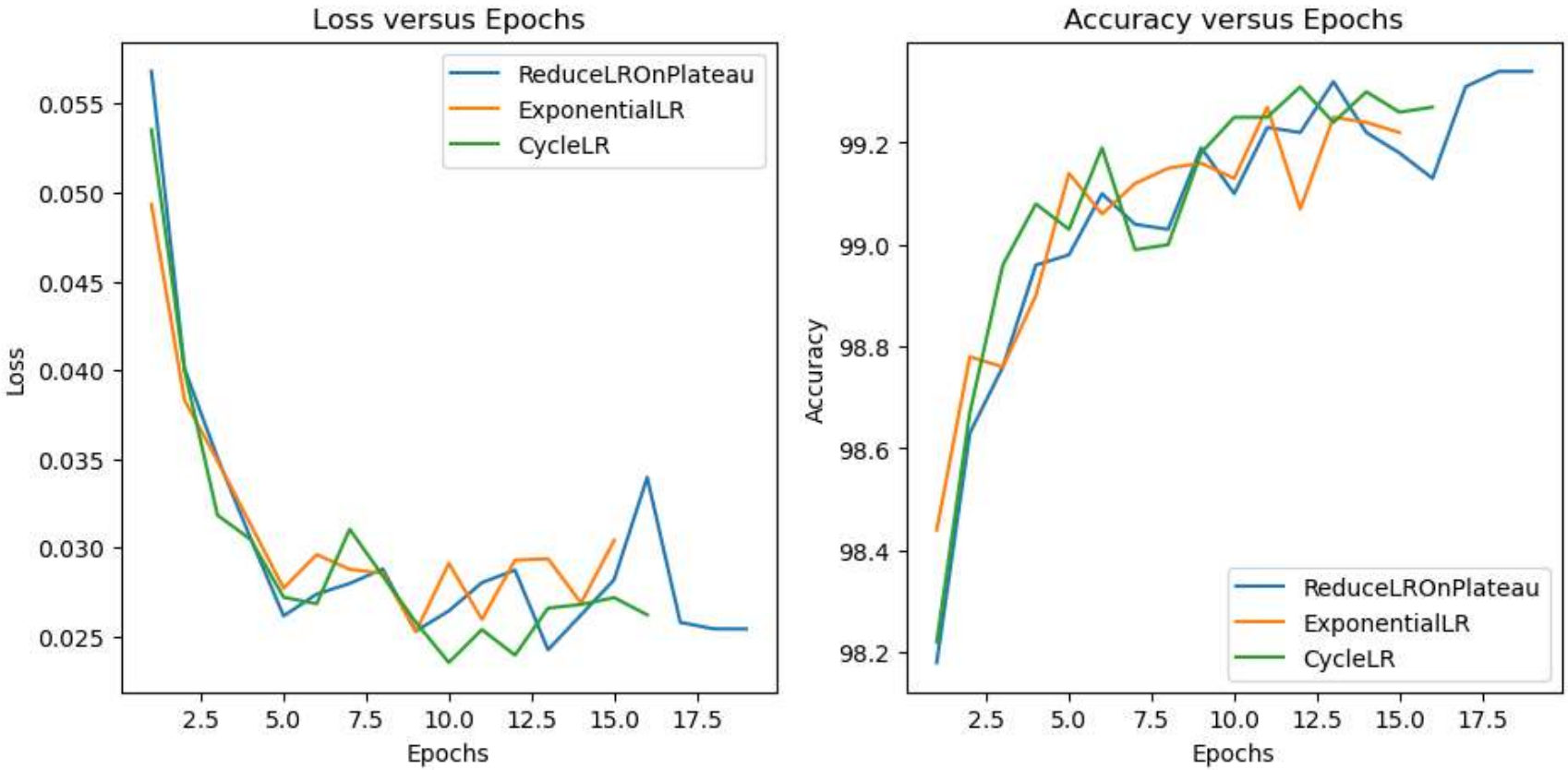
In [29]:
```
config = {
        'model' : 'basic',
        'dataset' : 'mnist',
        'batch_size' : 256,
        'num_workers' : 4,
        'lr' : 1e-3,
        'epochs' : 50,
```

```
Epoch 0.03: adjusting learning rate of group 0 to 1.5389e-03.
[INFO]: Epoch 6/50
Train loss: 0.0387, Train accuracy: 98.70
Validation loss: 0.0269, Validation accuracy: 99.19
------------------------------------------------
Epoch 0.03: adjusting learning rate of group 0 to 1.5316e-03.
[INFO]: Epoch 7/50
Train loss: 0.0364, Train accuracy: 98.84
Validation loss: 0.0310, Validation accuracy: 98.99
------------------------------------------------
Epoch 0.03: adjusting learning rate of group 0 to 1.6147e-03.
[INFO]: Epoch 8/50
Train loss: 0.0341, Train accuracy: 98.90
Validation loss: 0.0284, Validation accuracy: 99.00
------------------------------------------------
Epoch 0.03: adjusting learning rate of group 0 to 1.5627e-03.
[INFO]: Epoch 9/50
Train loss: 0.0308, Train accuracy: 99.02
Validation loss: 0.0258, Validation accuracy: 99.18
------------------------------------------------
Epoch 0.03: adjusting learning rate of group 0 to 1.5111e-03.
[INFO]: Epoch 10/50
Train loss: 0.0281, Train accuracy: 99.06
Validation loss: 0.0235, Validation accuracy: 99.25
------------------------------------------------
Epoch 0.02: adjusting learning rate of group 0 to 1.4662e-03.
[INFO]: Epoch 11/50
Train loss: 0.0244, Train accuracy: 99.20
Validation loss: 0.0254, Validation accuracy: 99.25
------------------------------------------------
Epoch 0.03: adjusting learning rate of group 0 to 1.5029e-03.
[INFO]: Epoch 12/50
Train loss: 0.0244, Train accuracy: 99.20
Validation loss: 0.0240, Validation accuracy: 99.31
------------------------------------------------
Epoch 0.02: adjusting learning rate of group 0 to 1.4744e-03.
[INFO]: Epoch 13/50
Train loss: 0.0238, Train accuracy: 99.17
Validation loss: 0.0266, Validation accuracy: 99.24
------------------------------------------------
Epoch 0.03: adjusting learning rate of group 0 to 1.5269e-03.
[INFO]: Epoch 14/50
Train loss: 0.0219, Train accuracy: 99.29
Validation loss: 0.0268, Validation accuracy: 99.30
------------------------------------------------
```

```
Epoch 0.03: adjusting learning rate of group 0 to 1.5309e-03.
[INFO]: Epoch 15/50
Train loss: 0.0208, Train accuracy: 99.32
Validation loss: 0.0272, Validation accuracy: 99.26
------------------------------------------------
Epoch 0.03: adjusting learning rate of group 0 to 1.5385e-03.
[INFO]: Epoch 16/50
Train loss: 0.0206, Train accuracy: 99.32
Validation loss: 0.0262, Validation accuracy: 99.27
------------------------------------------------
Epoch 0.03: adjusting learning rate of group 0 to 1.5193e-03.
[INFO]: Early stopping
```

In [30]:
```python
plt.figure(figsize=(11,5))

plt.subplot(121)
for i in range(len(learning_rate_schedulers)):
    plt.plot(np.arange(len(val_loss_arrays[i])) + 1,val_loss_arrays[i], label=f"{learning_rate_schedulers[i]}")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title("Loss versus Epochs")
plt.legend()

plt.subplot(122)
for i in range(len(learning_rate_schedulers)):
    plt.plot(np.arange(len(val_accuracy_arrays[i])) + 1,val_accuracy_arrays[i], label=f"{learning_rate_schedulers[i]}")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title("Accuracy versus Epochs")
plt.legend()
```

Out[30]:
```
<matplotlib.legend.Legend at 0x1e1a5b36690>
```



In [31]:
```
for i in range(len(learning_rate_schedulers)):
    print(f"Accuracy of above model using {learning_rate_schedulers[i]} : {max(val_accuracy_arrays[i])}")
```

```
Accuracy of above model using ReduceLROnPlateau : 99.34
Accuracy of above model using ExponentialLR : 99.27
Accuracy of above model using CycleLR : 99.31
```

Q1

How does changing in model affect the final accuracy:

Changing the model leads small change in accuracy Basic model being 99.13% and ResNet18 being 99.55%. This is because MNIST is a relatively simple dataset and doesnt not require the such a large network like ResNet18 to achieve high accuracies.

How does changing the learning rate scheuduler affect the final accuracy:

Changing the learning rate scheduler also does not boast a large difference in accuarcies, however ReduceLROnPlateau does seem to have a slight advantage over the other schedulers. [Shown in the graphs plotted in section 2.1.3]

How does changing the optimizer affect the final accuracy:

Changing the optimizer has the largest affect on final accuracy, this is due to the fact that Adam and RMSProp optimizer are adaptative optimizers and adjust the learning rate during training whereas SGD does not have any such adaptive capability. Thus Adam and RMSProp converge alot faster (require lesser epochs) than SGD, with Adam performing slightly better than RMSProp. [Shown in the graphs plotted in section 2.1.2]

CIFAR

BasicNet

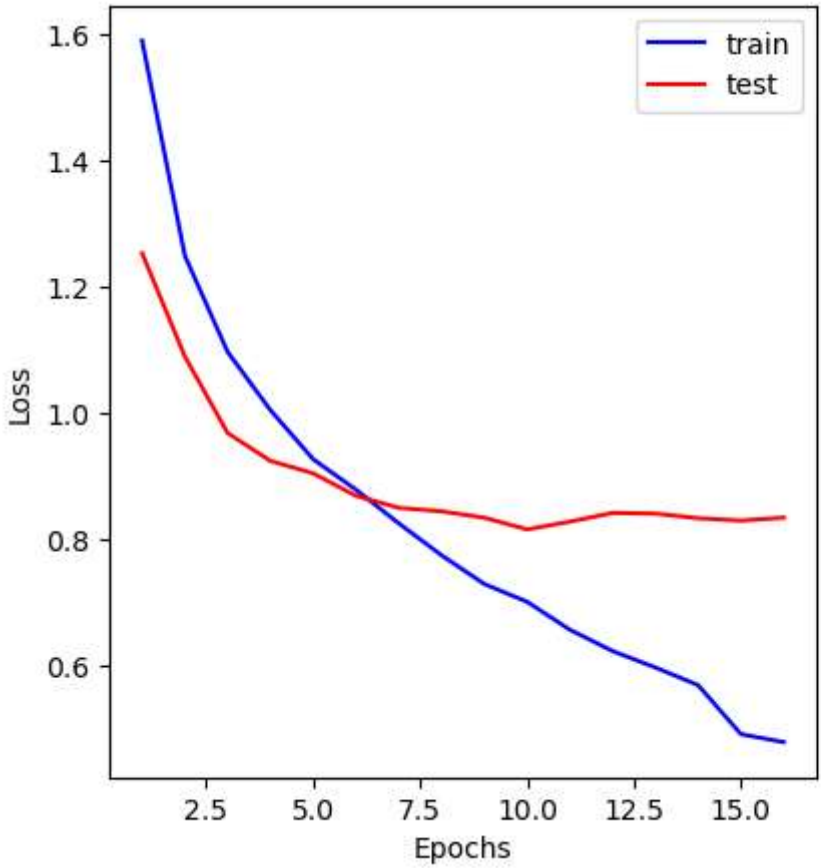
In [34]:
```
config = {
        'model' : 'basic',
        'dataset' : 'cifar10',
        'batch_size' : 256,
        'num_workers' : 4,
        'lr' : 1e-3,
        'epochs' : 50,
        'criterion' : nn.NLLLoss,
        'optimizer' : 'Adam',
        'device' : 'cuda' if torch.cuda.is_available() else 'cpu',
        'input_channels' : 3,
        'num_classes' : 10,
        'pin_memory' : True if torch.cuda.is_available() else False,
        'lr_scheduler' : "ReduceLROnPlateau"
    }

val_loss_array, val_accuracy_array, train_loss_array, train_accuracy_array, flops, params, inference_time = train_valid
```

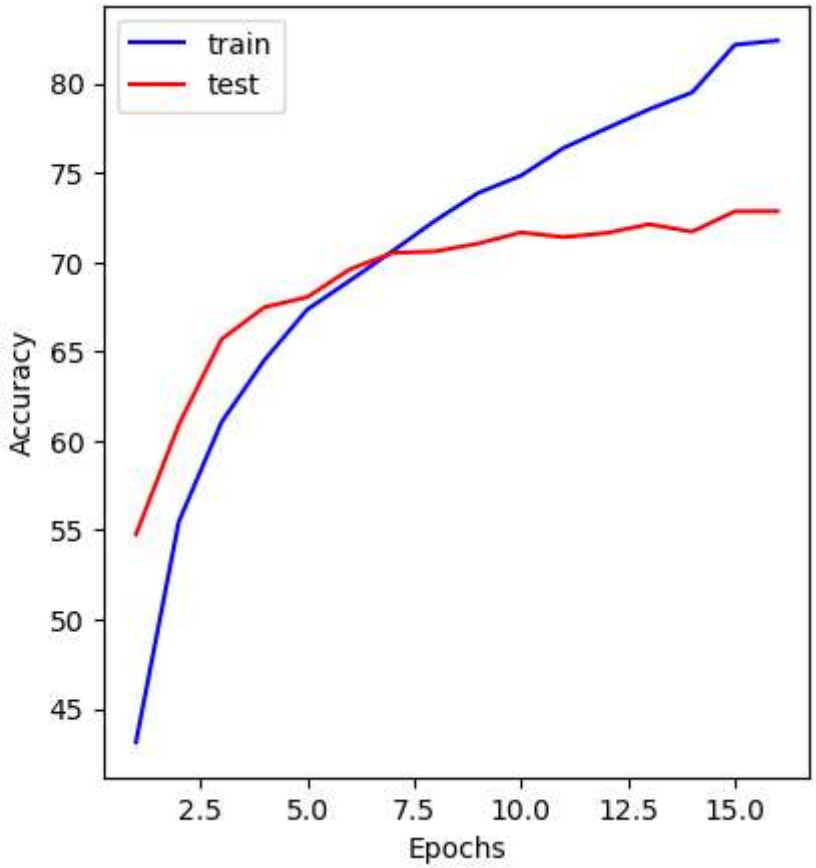
```
Files already downloaded and verified
[INFO]: Epoch 1/50
Train loss: 1.5898, Train accuracy: 43.14
Validation loss: 1.2528, Validation accuracy: 54.77
-------------------------------------------------
[INFO]: Epoch 2/50
Train loss: 1.2486, Train accuracy: 55.48
Validation loss: 1.0892, Validation accuracy: 60.90
-------------------------------------------------
[INFO]: Epoch 3/50
Train loss: 1.0963, Train accuracy: 61.05
Validation loss: 0.9677, Validation accuracy: 65.69
-------------------------------------------------
[INFO]: Epoch 4/50
Train loss: 1.0041, Train accuracy: 64.52
Validation loss: 0.9236, Validation accuracy: 67.47
-------------------------------------------------
[INFO]: Epoch 5/50
Train loss: 0.9261, Train accuracy: 67.34
Validation loss: 0.9039, Validation accuracy: 68.04
-------------------------------------------------
[INFO]: Epoch 6/50
Train loss: 0.8780, Train accuracy: 68.97
Validation loss: 0.8683, Validation accuracy: 69.60
-------------------------------------------------
[INFO]: Epoch 7/50
Train loss: 0.8246, Train accuracy: 70.62
Validation loss: 0.8491, Validation accuracy: 70.52
-------------------------------------------------
[INFO]: Epoch 8/50
Train loss: 0.7741, Train accuracy: 72.34
Validation loss: 0.8437, Validation accuracy: 70.60
-------------------------------------------------
[INFO]: Epoch 9/50
Train loss: 0.7285, Train accuracy: 73.87
Validation loss: 0.8335, Validation accuracy: 71.04
-------------------------------------------------
[INFO]: Epoch 10/50
Train loss: 0.7003, Train accuracy: 74.84
Validation loss: 0.8150, Validation accuracy: 71.66
-------------------------------------------------
[INFO]: Epoch 11/50
Train loss: 0.6560, Train accuracy: 76.40
Validation loss: 0.8272, Validation accuracy: 71.40
-------------------------------------------------
```

```
[INFO]: Epoch 12/50
Train loss: 0.6221, Train accuracy: 77.49
Validation loss: 0.8411, Validation accuracy: 71.63
--------------------------------------------------
[INFO]: Epoch 13/50
Train loss: 0.5961, Train accuracy: 78.56
Validation loss: 0.8401, Validation accuracy: 72.12
--------------------------------------------------
[INFO]: Epoch 14/50
Train loss: 0.5681, Train accuracy: 79.49
Validation loss: 0.8327, Validation accuracy: 71.70
--------------------------------------------------
Epoch 00014: reducing learning rate of group 0 to 5.0000e-05.
[INFO]: Epoch 15/50
Train loss: 0.4904, Train accuracy: 82.17
Validation loss: 0.8290, Validation accuracy: 72.84
--------------------------------------------------
[INFO]: Epoch 16/50
Train loss: 0.4779, Train accuracy: 82.40
Validation loss: 0.8337, Validation accuracy: 72.85
--------------------------------------------------
[INFO]: Early stopping
```

In [35]:
```
plt.figure(figsize=(10,5))

plt.subplot(121)
plt.plot(np.arange(len(train_loss_array)) + 1,train_loss_array,c='b',label='train')
plt.plot(np.arange(len(train_loss_array)) + 1,val_loss_array, c='r',label='test')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(122)
plt.plot(np.arange(len(train_loss_array)) + 1,train_accuracy_array,c='b',label='train')
plt.plot(np.arange(len(train_loss_array)) + 1,val_accuracy_array, c='r',label='test')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```

Out[35]:
```
<matplotlib.legend.Legend at 0x1e18257cf50>
```





In [36]:
```
print(f"The Accuracy of the above model on test data is : {max(val_accuracy_array)}")
```

```
The Accuracy of the above model on test data is : 72.85
```

Resnet 18 base

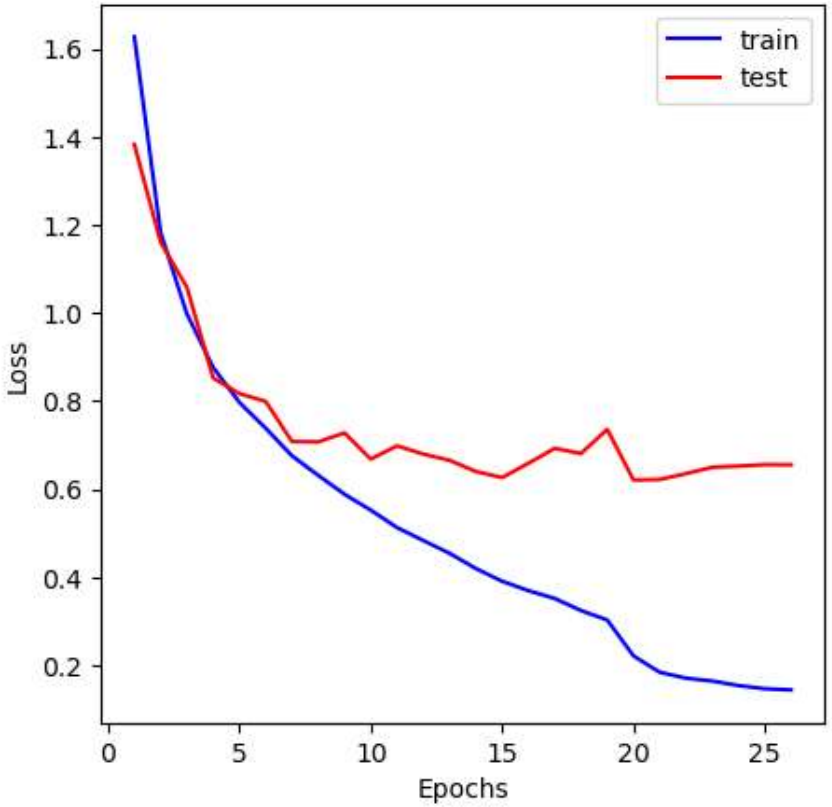
In [37]:
```
config = {
        'model' : 'resnet18',
        'dataset' : 'cifar10',
        'batch_size' : 256,
        'num_workers' : 4,
        'lr' : 1e-3,
        'epochs' : 50,
        'criterion' : nn.NLLLoss,
        'optimizer' : 'Adam',
```

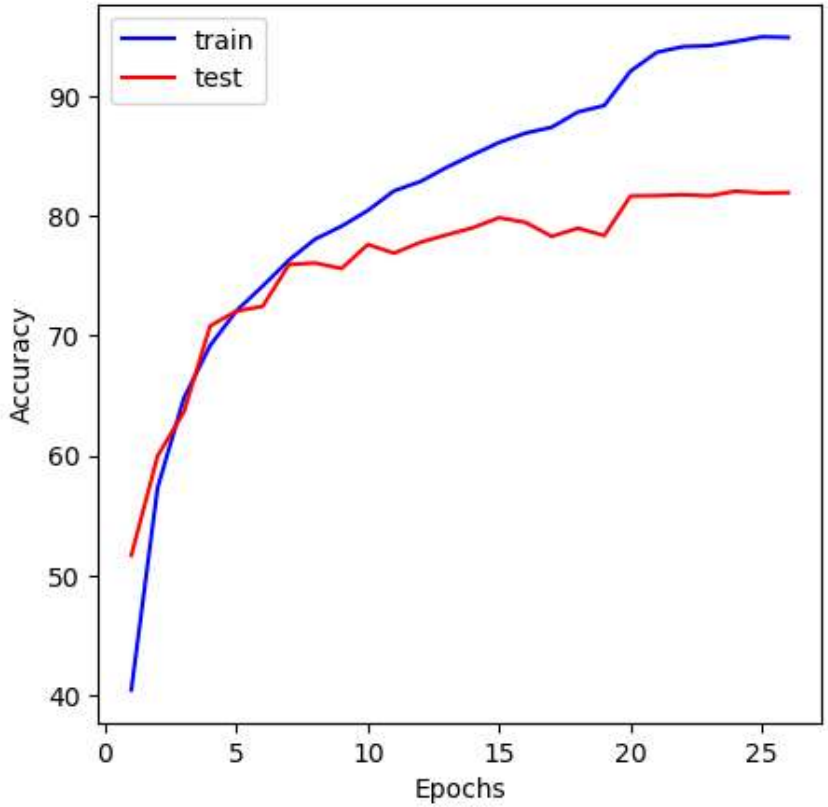
```
[INFO]: Epoch 12/50
Train loss: 0.4834, Train accuracy: 82.88
Validation loss: 0.6795, Validation accuracy: 77.80
------------------------------------------------
[INFO]: Epoch 13/50
Train loss: 0.4544, Train accuracy: 84.06
Validation loss: 0.6656, Validation accuracy: 78.43
------------------------------------------------
[INFO]: Epoch 14/50
Train loss: 0.4199, Train accuracy: 85.12
Validation loss: 0.6402, Validation accuracy: 79.02
------------------------------------------------
[INFO]: Epoch 15/50
Train loss: 0.3910, Train accuracy: 86.14
Validation loss: 0.6264, Validation accuracy: 79.86
------------------------------------------------
[INFO]: Epoch 16/50
Train loss: 0.3696, Train accuracy: 86.92
Validation loss: 0.6589, Validation accuracy: 79.47
------------------------------------------------
[INFO]: Epoch 17/50
Train loss: 0.3519, Train accuracy: 87.41
Validation loss: 0.6928, Validation accuracy: 78.30
------------------------------------------------
[INFO]: Epoch 18/50
Train loss: 0.3246, Train accuracy: 88.69
Validation loss: 0.6809, Validation accuracy: 78.98
------------------------------------------------
[INFO]: Epoch 19/50
Train loss: 0.3031, Train accuracy: 89.22
Validation loss: 0.7359, Validation accuracy: 78.37
------------------------------------------------
Epoch 00019: reducing learning rate of group 0 to 5.0000e-05.
[INFO]: Epoch 20/50
Train loss: 0.2215, Train accuracy: 92.10
Validation loss: 0.6207, Validation accuracy: 81.67
------------------------------------------------
[INFO]: Epoch 21/50
Train loss: 0.1844, Train accuracy: 93.67
Validation loss: 0.6221, Validation accuracy: 81.70
------------------------------------------------
[INFO]: Epoch 22/50
Train loss: 0.1710, Train accuracy: 94.14
Validation loss: 0.6354, Validation accuracy: 81.78
------------------------------------------------
```

```
[INFO]: Epoch 23/50
Train loss: 0.1644, Train accuracy: 94.22
Validation loss: 0.6498, Validation accuracy: 81.68
-------------------------------------------------
[INFO]: Epoch 24/50
Train loss: 0.1542, Train accuracy: 94.57
Validation loss: 0.6524, Validation accuracy: 82.07
-------------------------------------------------
Epoch 00024: reducing learning rate of group 0 to 2.5000e-06.
[INFO]: Epoch 25/50
Train loss: 0.1470, Train accuracy: 94.96
Validation loss: 0.6557, Validation accuracy: 81.92
-------------------------------------------------
[INFO]: Epoch 26/50
Train loss: 0.1444, Train accuracy: 94.91
Validation loss: 0.6554, Validation accuracy: 81.95
-------------------------------------------------
[INFO]: Early stopping
```

In [38]:
```
plt.figure(figsize=(11,5))

plt.subplot(121)
plt.plot(np.arange(len(train_loss_array)) + 1,train_loss_array,c='b',label='train')
plt.plot(np.arange(len(train_loss_array)) + 1,val_loss_array, c='r',label='test')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(122)
plt.plot(np.arange(len(train_loss_array)) + 1,train_accuracy_array,c='b',label='train')
plt.plot(np.arange(len(train_loss_array)) + 1,val_accuracy_array, c='r',label='test')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```

Out[38]:
```
<matplotlib.legend.Legend at 0x1e1a5b35710>
```





In [39]:
```
print(f"The Accuracy of the above model on test data is : {max(val_accuracy_array)}")
```

```
The Accuracy of the above model on test data is : 82.07
```

Hyperparameter Search

Optimizer

In [40]:
```
config = {
        'model' : 'resnet18',
        'dataset' : 'cifar10',
        'batch_size' : 256,
        'num_workers' : 4,
        'lr' : 1e-3,
        'epochs' : 50,
        'criterion' : nn.NLLLoss,
```

```
        'optimizer' : 'SGD',
        'device' : 'cuda' if torch.cuda.is_available() else 'cpu',
        'input_channels' : 3,
        'num_classes' : 10,
        'pin_memory' : True if torch.cuda.is_available() else False,
        'lr_scheduler' : "ReduceLROnPlateau"
    }

optimizers = ['Adam', 'SGD', 'RMSprop',]

val_loss_arrays, val_accuracy_arrays, train_loss_arrays, train_accuracy_arrays = [], [], [], []

for optimizer in optimizers:
    config['optimizer'] = optimizer
    val_loss_array, val_accuracy_array, train_loss_array, train_accuracy_array, flops, params, inference_time = train_v

    val_loss_arrays.append(val_loss_array)
    val_accuracy_arrays.append(val_accuracy_array)
```

```
------------------------------------------------
[INFO]: Epoch 7/50
Train loss: 0.7175, Train accuracy: 74.87
Validation loss: 0.8857, Validation accuracy: 70.10
------------------------------------------------
[INFO]: Epoch 8/50
Train loss: 0.6682, Train accuracy: 76.51
Validation loss: 1.0752, Validation accuracy: 65.74
------------------------------------------------
[INFO]: Epoch 9/50
Train loss: 0.6301, Train accuracy: 77.90
Validation loss: 0.7901, Validation accuracy: 73.48
------------------------------------------------
[INFO]: Epoch 10/50
Train loss: 0.5809, Train accuracy: 79.56
Validation loss: 5.5227, Validation accuracy: 49.61
------------------------------------------------
[INFO]: Epoch 11/50
Train loss: 0.5442, Train accuracy: 80.88
Validation loss: 0.7588, Validation accuracy: 75.29
------------------------------------------------
[INFO]: Epoch 12/50
Train loss: 0.5011, Train accuracy: 82.44
Validation loss: 0.8091, Validation accuracy: 74.38
------------------------------------------------
[INFO]: Epoch 13/50
Train loss: 0.4747, Train accuracy: 83.20
Validation loss: 0.7680, Validation accuracy: 74.67
------------------------------------------------
[INFO]: Epoch 14/50
Train loss: 0.4429, Train accuracy: 84.36
Validation loss: 0.8918, Validation accuracy: 71.93
------------------------------------------------
[INFO]: Epoch 15/50
Train loss: 0.4186, Train accuracy: 85.08
Validation loss: 0.8037, Validation accuracy: 75.29
------------------------------------------------
Epoch 00015: reducing learning rate of group 0 to 5.0000e-05.
[INFO]: Epoch 16/50
Train loss: 0.3262, Train accuracy: 88.56
Validation loss: 0.6289, Validation accuracy: 80.35
------------------------------------------------
[INFO]: Epoch 17/50
Train loss: 0.2851, Train accuracy: 89.95
Validation loss: 0.6434, Validation accuracy: 80.46
```

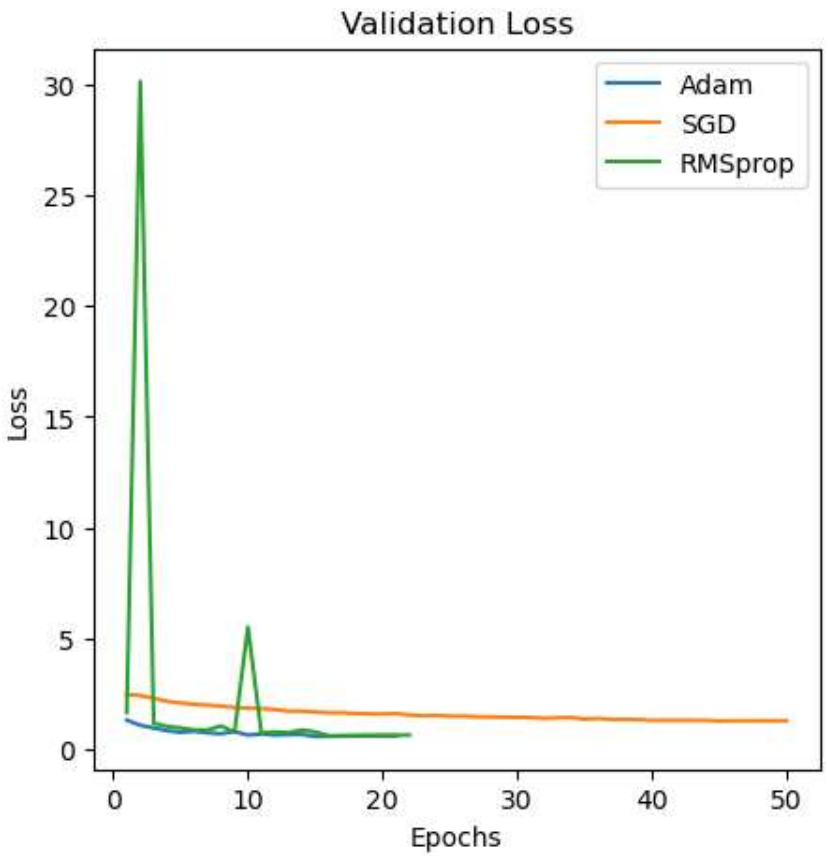
```
------------------------------------------------
[INFO]: Epoch 18/50
Train loss: 0.2697, Train accuracy: 90.53
Validation loss: 0.6436, Validation accuracy: 80.66
------------------------------------------------
[INFO]: Epoch 19/50
Train loss: 0.2571, Train accuracy: 90.78
Validation loss: 0.6564, Validation accuracy: 80.77
------------------------------------------------
[INFO]: Epoch 20/50
Train loss: 0.2512, Train accuracy: 91.18
Validation loss: 0.6568, Validation accuracy: 80.77
------------------------------------------------
Epoch 00020: reducing learning rate of group 0 to 2.5000e-06.
[INFO]: Epoch 21/50
Train loss: 0.2410, Train accuracy: 91.41
Validation loss: 0.6551, Validation accuracy: 80.83
------------------------------------------------
[INFO]: Epoch 22/50
Train loss: 0.2416, Train accuracy: 91.37
Validation loss: 0.6645, Validation accuracy: 80.84
------------------------------------------------
[INFO]: Early stopping
```

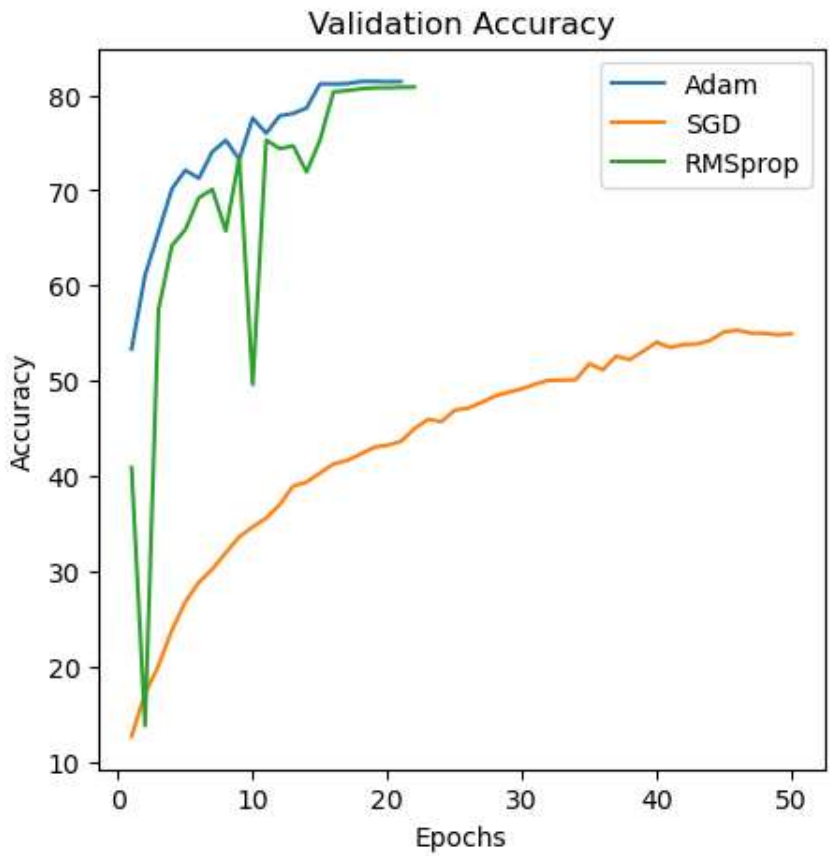
In [41]:

```
plt.figure(figsize=(11,5))

plt.subplot(121)
for i in range(len(optimizers)):
    plt.plot(np.arange(len(val_loss_arrays[i])) + 1,val_loss_arrays[i], label=f"{optimizers[i]}")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title("Validation Loss")
plt.legend()

plt.subplot(122)
for i in range(len(optimizers)):
    plt.plot(np.arange(len(val_accuracy_arrays[i])) + 1,val_accuracy_arrays[i], label=f"{optimizers[i]}")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title("Validation Accuracy")
plt.legend()
```

Out[41]:

```
<matplotlib.legend.Legend at 0x1e1828a8210>
```





In [42]:
```
for i in range(len(optimizers)):
    print(f"Accuracy of above model using {optimizers[i]} : {max(val_accuracy_arrays[i])}")
```

```
Accuracy of above model using Adam : 81.44
Accuracy of above model using SGD : 55.32
Accuracy of above model using RMSprop : 80.84
```

Learning Rate

In [43]:
```
config = {
        'model' : 'resnet18',
        'dataset' : 'cifar10',
        'batch_size' : 256,
        'num_workers' : 4,
        'lr' : 1e-3,
        'epochs' : 100,
```

```
        'criterion' : nn.NLLLoss,
        'optimizer' : 'Adam',
        'device' : 'cuda' if torch.cuda.is_available() else 'cpu',
        'input_channels' : 3,
        'num_classes' : 10,
        'pin_memory' : True if torch.cuda.is_available() else False,
        'lr_scheduler' : "ReduceLROnPlateau"
    }

learning_rates = [1e-5,1e-4,1e-3,1e-2,1e-1]

val_loss_arrays, val_accuracy_arrays, train_loss_arrays, train_accuracy_arrays = [], [], [], []

for lr in learning_rates:
    config['lr'] = lr
    val_loss_array, val_accuracy_array, train_loss_array, train_accuracy_array, flops, params, inference_time = train_v

    val_loss_arrays.append(val_loss_array)
    val_accuracy_arrays.append(val_accuracy_array)
```

```
Files already downloaded and verified
[INFO]: Epoch 1/100
Train loss: 2.4003, Train accuracy: 13.19
Validation loss: 2.5348, Validation accuracy: 10.44
-------------------------------------------------
[INFO]: Epoch 2/100
Train loss: 2.2281, Train accuracy: 17.88
Validation loss: 2.4411, Validation accuracy: 15.45
-------------------------------------------------
[INFO]: Epoch 3/100
Train loss: 2.0999, Train accuracy: 22.20
Validation loss: 2.2998, Validation accuracy: 20.47
-------------------------------------------------
[INFO]: Epoch 4/100
Train loss: 1.9950, Train accuracy: 25.84
Validation loss: 2.2208, Validation accuracy: 24.00
-------------------------------------------------
[INFO]: Epoch 5/100
Train loss: 1.9194, Train accuracy: 28.24
Validation loss: 2.1435, Validation accuracy: 26.90
-------------------------------------------------
[INFO]: Epoch 6/100
Train loss: 1.8653, Train accuracy: 30.52
Validation loss: 2.0425, Validation accuracy: 30.56
-------------------------------------------------
[INFO]: Epoch 7/100
Train loss: 1.8190, Train accuracy: 32.24
Validation loss: 1.9057, Validation accuracy: 33.70
-------------------------------------------------
[INFO]: Epoch 8/100
Train loss: 1.7787, Train accuracy: 33.66
Validation loss: 1.9116, Validation accuracy: 34.71
-------------------------------------------------
[INFO]: Epoch 9/100
Train loss: 1.7459, Train accuracy: 34.88
Validation loss: 1.8561, Validation accuracy: 36.07
-------------------------------------------------
[INFO]: Epoch 10/100
Train loss: 1.7139, Train accuracy: 36.02
Validation loss: 1.8090, Validation accuracy: 37.22
-------------------------------------------------
[INFO]: Epoch 11/100
Train loss: 1.6853, Train accuracy: 37.18
Validation loss: 1.7573, Validation accuracy: 38.96
-------------------------------------------------
```

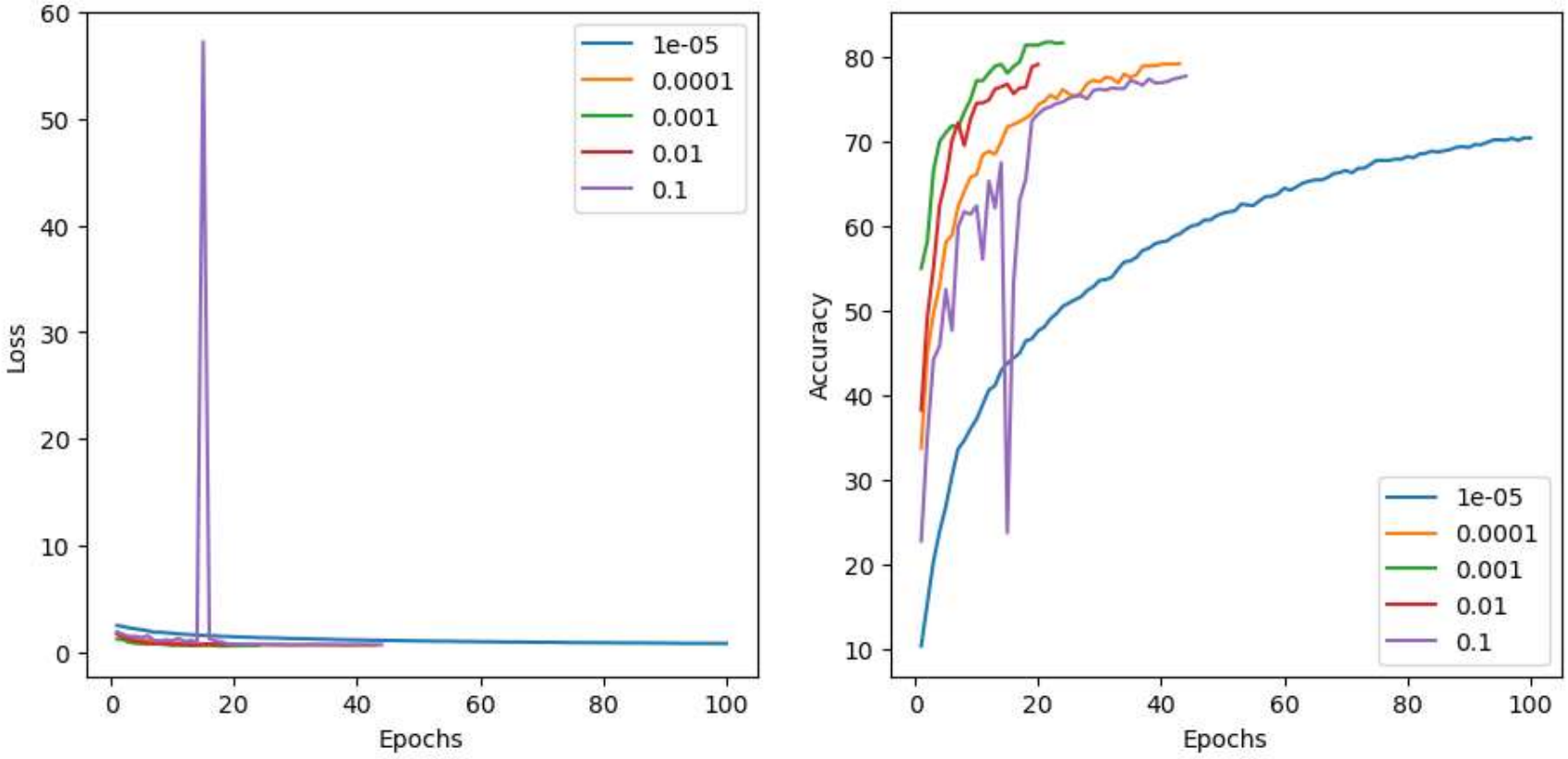
```
Validation loss: 0.7328, Validation accuracy: 77.23
------------------------------------------------
[INFO]: Epoch 36/100
Train loss: 0.5191, Train accuracy: 82.21
Validation loss: 0.7389, Validation accuracy: 76.99
------------------------------------------------
[INFO]: Epoch 37/100
Train loss: 0.5144, Train accuracy: 82.28
Validation loss: 0.7479, Validation accuracy: 76.68
------------------------------------------------
[INFO]: Epoch 38/100
Train loss: 0.5014, Train accuracy: 82.66
Validation loss: 0.7297, Validation accuracy: 77.42
------------------------------------------------
[INFO]: Epoch 39/100
Train loss: 0.4956, Train accuracy: 82.78
Validation loss: 0.7472, Validation accuracy: 76.94
------------------------------------------------
[INFO]: Epoch 40/100
Train loss: 0.4779, Train accuracy: 83.37
Validation loss: 0.7492, Validation accuracy: 76.96
------------------------------------------------
[INFO]: Epoch 41/100
Train loss: 0.4638, Train accuracy: 84.13
Validation loss: 0.7558, Validation accuracy: 77.10
------------------------------------------------
[INFO]: Epoch 42/100
Train loss: 0.4572, Train accuracy: 84.24
Validation loss: 0.7448, Validation accuracy: 77.40
------------------------------------------------
Epoch 00042: reducing learning rate of group 0 to 2.5000e-04.
[INFO]: Epoch 43/100
Train loss: 0.4278, Train accuracy: 85.28
Validation loss: 0.7393, Validation accuracy: 77.56
------------------------------------------------
[INFO]: Epoch 44/100
Train loss: 0.4182, Train accuracy: 85.59
Validation loss: 0.7411, Validation accuracy: 77.75
------------------------------------------------
[INFO]: Early stopping
```

In [44]:

```
plt.figure(figsize=(11,5))

plt.subplot(121)
for i in range(len(learning_rates)):
```

```
    plt.plot(np.arange(len(val_loss_arrays[i])) + 1,val_loss_arrays[i], label=f"{learning_rates[i]}")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(122)
for i in range(len(learning_rates)):
    plt.plot(np.arange(len(val_accuracy_arrays[i])) + 1,val_accuracy_arrays[i], label=f"{learning_rates[i]}")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```

Out[44]:
```
<matplotlib.legend.Legend at 0x1e175c5c490>
```



In [45]:
```
for i in range(len(learning_rates)):
    print(f"Accuracy of above model using LR={learning_rates[i]} : {max(val_accuracy_arrays[i])}")
```

```
Accuracy of above model using LR=1e-05 : 70.45
Accuracy of above model using LR=0.0001 : 79.22
Accuracy of above model using LR=0.001 : 81.8
Accuracy of above model using LR=0.01 : 79.13
Accuracy of above model using LR=0.1 : 77.75
```

Learning Rate Scheduler

In [46]:

```python
config = {
        'model' : 'resnet18',
        'dataset' : 'cifar10',
        'batch_size' : 256,
        'num_workers' : 4,
        'lr' : 1e-3,
        'epochs' : 50,
        'criterion' : nn.NLLLoss,
        'optimizer' : 'Adam',
        'device' : 'cuda' if torch.cuda.is_available() else 'cpu',
        'input_channels' : 3,
        'num_classes' : 10,
        'pin_memory' : True if torch.cuda.is_available() else False,
        'lr_scheduler' : "ReduceLROnPlateau"
    }

learning_rate_schedulers = ["ReduceLROnPlateau", "ExponentialLR", "CycleLR"]

val_loss_arrays, val_accuracy_arrays, train_loss_arrays, train_accuracy_arrays = [], [], [], []

for lr in learning_rate_schedulers:
    config['lr_scheduler'] = lr
    val_loss_array, val_accuracy_array, train_loss_array, train_accuracy_array, flops, params, inference_time = train_v

    val_loss_arrays.append(val_loss_array)
    val_accuracy_arrays.append(val_accuracy_array)
```

```
Files already downloaded and verified
[INFO]: Epoch 1/50
Train loss: 1.6173, Train accuracy: 40.68
Validation loss: 1.3420, Validation accuracy: 51.94
-------------------------------------------------
[INFO]: Epoch 2/50
Train loss: 1.1986, Train accuracy: 56.84
Validation loss: 1.0329, Validation accuracy: 62.87
-------------------------------------------------
[INFO]: Epoch 3/50
Train loss: 1.0049, Train accuracy: 64.45
Validation loss: 1.0241, Validation accuracy: 64.56
-------------------------------------------------
[INFO]: Epoch 4/50
Train loss: 0.8923, Train accuracy: 68.50
Validation loss: 0.8615, Validation accuracy: 70.02
-------------------------------------------------
[INFO]: Epoch 5/50
Train loss: 0.8134, Train accuracy: 71.42
Validation loss: 0.8410, Validation accuracy: 70.62
-------------------------------------------------
[INFO]: Epoch 6/50
Train loss: 0.7446, Train accuracy: 73.71
Validation loss: 0.8282, Validation accuracy: 70.65
-------------------------------------------------
[INFO]: Epoch 7/50
Train loss: 0.6871, Train accuracy: 76.05
Validation loss: 0.8174, Validation accuracy: 72.43
-------------------------------------------------
[INFO]: Epoch 8/50
Train loss: 0.6385, Train accuracy: 77.46
Validation loss: 0.7921, Validation accuracy: 73.55
-------------------------------------------------
[INFO]: Epoch 9/50
Train loss: 0.5991, Train accuracy: 78.89
Validation loss: 0.6867, Validation accuracy: 76.39
-------------------------------------------------
[INFO]: Epoch 10/50
Train loss: 0.5543, Train accuracy: 80.46
Validation loss: 0.7583, Validation accuracy: 74.88
-------------------------------------------------
[INFO]: Epoch 11/50
Train loss: 0.5189, Train accuracy: 81.58
Validation loss: 0.6857, Validation accuracy: 77.08
-------------------------------------------------
```

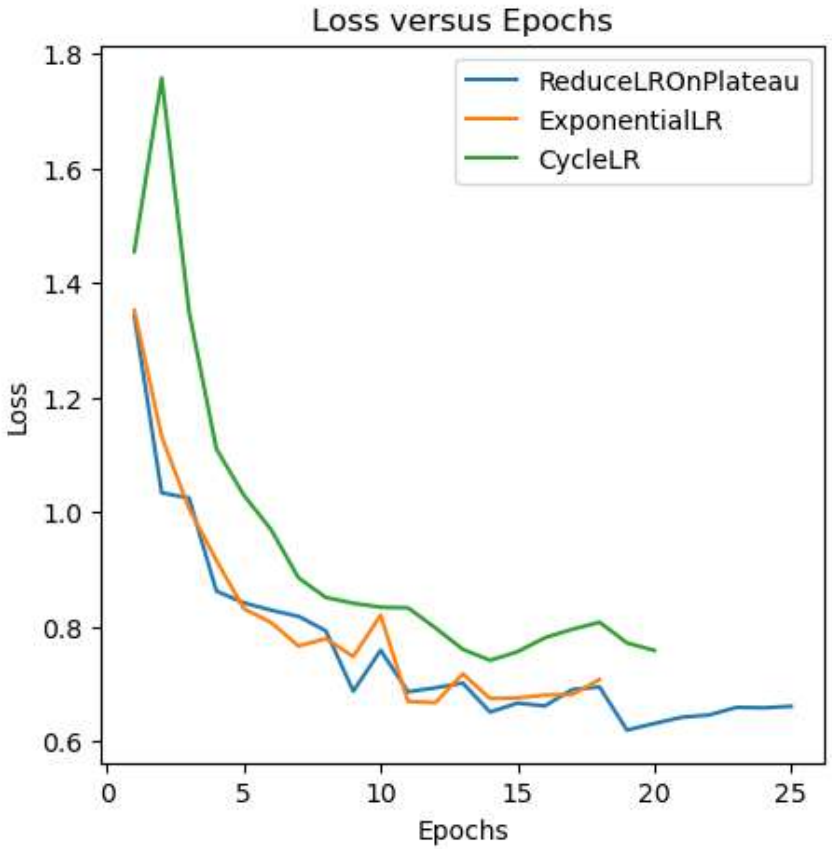
```
Train loss: 0.5178, Train accuracy: 82.08
Validation loss: 0.7556, Validation accuracy: 75.87
------------------------------------------------
Epoch 0.76: adjusting learning rate of group 0 to 1.5960e-02.
[INFO]: Epoch 16/50
Train loss: 0.5018, Train accuracy: 82.32
Validation loss: 0.7800, Validation accuracy: 74.97
------------------------------------------------
Epoch 0.78: adjusting learning rate of group 0 to 1.6443e-02.
[INFO]: Epoch 17/50
Train loss: 0.4918, Train accuracy: 82.69
Validation loss: 0.7947, Validation accuracy: 74.66
------------------------------------------------
Epoch 0.79: adjusting learning rate of group 0 to 1.6736e-02.
[INFO]: Epoch 18/50
Train loss: 0.4720, Train accuracy: 83.68
Validation loss: 0.8072, Validation accuracy: 74.63
------------------------------------------------
Epoch 0.81: adjusting learning rate of group 0 to 1.6983e-02.
[INFO]: Epoch 19/50
Train loss: 0.4572, Train accuracy: 84.07
Validation loss: 0.7710, Validation accuracy: 75.79
------------------------------------------------
Epoch 0.77: adjusting learning rate of group 0 to 1.6267e-02.
[INFO]: Epoch 20/50
Train loss: 0.4386, Train accuracy: 84.69
Validation loss: 0.7578, Validation accuracy: 75.81
------------------------------------------------
Epoch 0.76: adjusting learning rate of group 0 to 1.6005e-02.
[INFO]: Early stopping
```

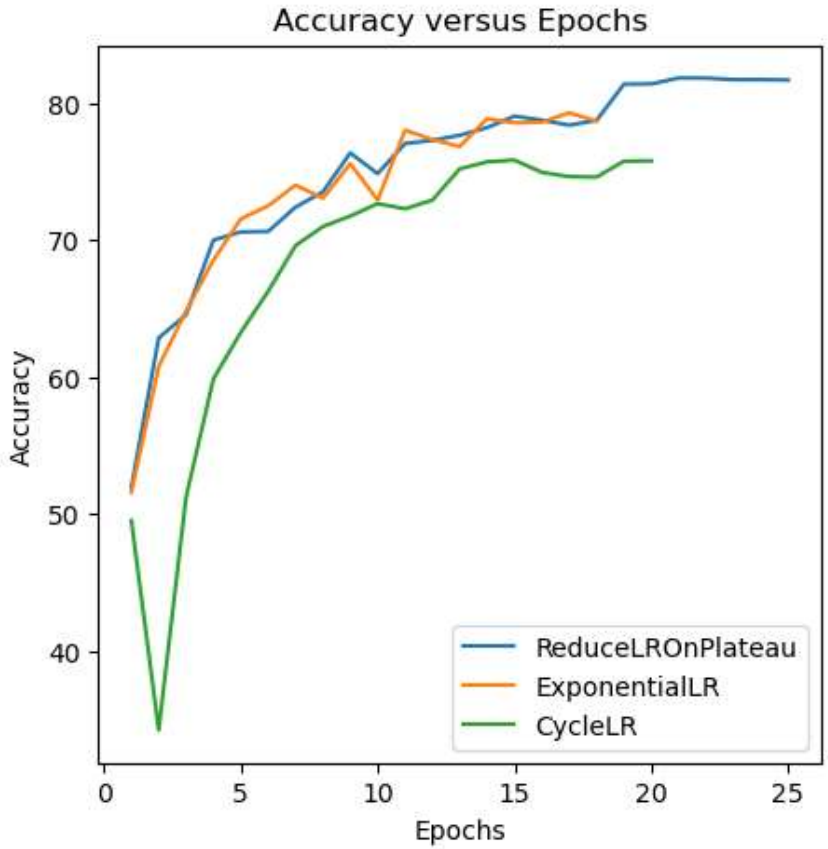
In [47]:

```python
plt.figure(figsize=(11,5))

plt.subplot(121)
for i in range(len(learning_rate_schedulers)):
    plt.plot(np.arange(len(val_loss_arrays[i])) + 1,val_loss_arrays[i], label=f"{learning_rate_schedulers[i]}")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title("Loss versus Epochs")
plt.legend()

plt.subplot(122)
for i in range(len(learning_rate_schedulers)):
    plt.plot(np.arange(len(val_accuracy_arrays[i])) + 1,val_accuracy_arrays[i], label=f"{learning_rate_schedulers[i]}")
plt.xlabel('Epochs')
```

```
plt.ylabel('Accuracy')
plt.title("Accuracy versus Epochs")
plt.legend()
```

Out[47]: <matplotlib.legend.Legend at 0x1e12a4ae690>





In [48]:
```
for i in range(len(learning_rate_schedulers)):
    print(f"Accuracy of above model using {learning_rate_schedulers[i]} : {max(val_accuracy_arrays[i])}")
```

```
Accuracy of above model using ReduceLROnPlateau : 81.87
Accuracy of above model using ExponentialLR : 79.32
Accuracy of above model using CycleLR : 75.87
```

Q2

Observations during tuning of Learning Rate Schedules and Optimizers:

During tuning of the learning rate schedulers, here ReduceLROnPlateau and ExponentialLR slightly edged the CycleLR, with ReduceLROnPlateau performing slightly better than ExponentialLR. With respect to the optimizers, the result were similar to that of MNIST, where Adam and RMSProp converging alot faster than SGD. [Shown in plot in Section 2.4.1 and 2.4.3]

Compared to MNIST how are the learning rates?

Comparing many different learning rates, a learning rate of 1e-3 was found to be optimal for training. This is similar to that of MNIST as well. [Shown in plot in Section 2.4.2]

Number of Epochs and Results?

The number of epochs required to train CIFAR10 was larger (50 compared to 30) than MNIST, this is likely due to the fact that it contains a larger amount of data (3x32x32 versus 1x28x28) due to a larger image and use of RGB images. Furthermore due to the added complexity, the best model turned out to be a ResNet18 model which achieved an accuracy of 82.07. Looking at the training graphs we can also see that the model was unable to overfit to the training data as well, thus it is possible that a ResNet18 model is not enough to capture the complexity of the dataset, at the same time other techniques can also be tried such as data augmentation to artifically improve the variety of samples in the training data. This tecniques can help improve the final validation accuracy.

How did you find the appropriate Hyperparameters

To find the appropriate hyperparameters, one hyperparameter was chosen and varied while other were fixed. This was due to the computational constraints. Provided a larger amount of computing power, an ideal method would be to randomly sample (or use some other types of sophisticated sampling strategies) from the set of all combinations of hyperparameters.

Task 3

Base ResNet18 with some layers from MobileNet

In [49]:
```
config = {
        'model' : 'resnet18_modified',
        'dataset' : 'cifar10',
        'batch_size' : 256,
        'num_workers' : 4,
        'lr' : 1e-3,
        'epochs' : 50,
```

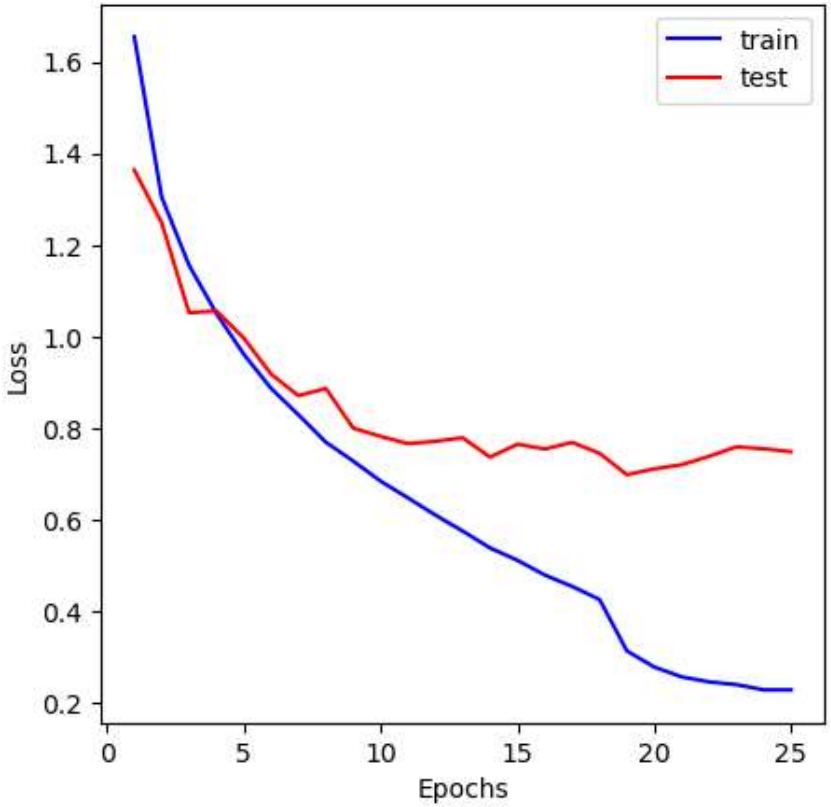
```
        'criterion' : nn.NLLLoss,
        'optimizer' : 'Adam',
        'device' : 'cuda' if torch.cuda.is_available() else 'cpu',
        'input_channels' : 3,
        'num_classes' : 10,
        'pin_memory' : True if torch.cuda.is_available() else False,
        'depthwise_convs': [True, False, False, False],
        'lr_scheduler' : "ReduceLROnPlateau"
    }

val_loss_array, val_accuracy_array, train_loss_array, train_accuracy_array, flops, params, inference_time = train_valid
```

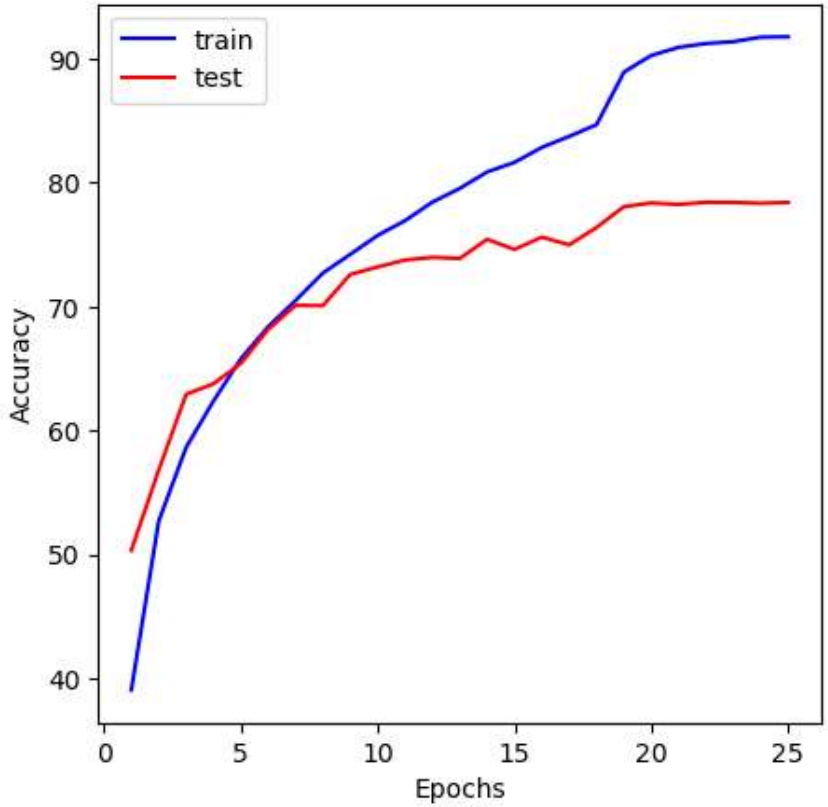
```
Files already downloaded and verified
[INFO]: Epoch 1/50
Train loss: 1.6541, Train accuracy: 39.06
Validation loss: 1.3635, Validation accuracy: 50.34
-------------------------------------------------
[INFO]: Epoch 2/50
Train loss: 1.3045, Train accuracy: 52.68
Validation loss: 1.2484, Validation accuracy: 56.81
-------------------------------------------------
[INFO]: Epoch 3/50
Train loss: 1.1558, Train accuracy: 58.63
Validation loss: 1.0522, Validation accuracy: 62.92
-------------------------------------------------
[INFO]: Epoch 4/50
Train loss: 1.0506, Train accuracy: 62.38
Validation loss: 1.0560, Validation accuracy: 63.77
-------------------------------------------------
[INFO]: Epoch 5/50
Train loss: 0.9616, Train accuracy: 65.78
Validation loss: 0.9970, Validation accuracy: 65.42
-------------------------------------------------
[INFO]: Epoch 6/50
Train loss: 0.8870, Train accuracy: 68.38
Validation loss: 0.9179, Validation accuracy: 68.17
-------------------------------------------------
[INFO]: Epoch 7/50
Train loss: 0.8294, Train accuracy: 70.49
Validation loss: 0.8713, Validation accuracy: 70.10
-------------------------------------------------
[INFO]: Epoch 8/50
Train loss: 0.7699, Train accuracy: 72.71
Validation loss: 0.8870, Validation accuracy: 70.08
-------------------------------------------------
[INFO]: Epoch 9/50
Train loss: 0.7286, Train accuracy: 74.22
Validation loss: 0.8005, Validation accuracy: 72.60
-------------------------------------------------
[INFO]: Epoch 10/50
Train loss: 0.6848, Train accuracy: 75.75
Validation loss: 0.7824, Validation accuracy: 73.20
-------------------------------------------------
[INFO]: Epoch 11/50
Train loss: 0.6484, Train accuracy: 76.94
Validation loss: 0.7666, Validation accuracy: 73.76
-------------------------------------------------
```

```
[INFO]: Epoch 23/50
Train loss: 0.2411, Train accuracy: 91.39
Validation loss: 0.7597, Validation accuracy: 78.41
-------------------------------------------------
Epoch 00023: reducing learning rate of group 0 to 2.5000e-06.
[INFO]: Epoch 24/50
Train loss: 0.2297, Train accuracy: 91.75
Validation loss: 0.7554, Validation accuracy: 78.34
-------------------------------------------------
[INFO]: Epoch 25/50
Train loss: 0.2296, Train accuracy: 91.78
Validation loss: 0.7490, Validation accuracy: 78.41
-------------------------------------------------
[INFO]: Early stopping
```

In [50]:
```
plt.figure(figsize=(11,5))

plt.subplot(121)
plt.plot(np.arange(len(train_loss_array)) + 1,train_loss_array,c='b',label='train')
plt.plot(np.arange(len(train_loss_array)) + 1,val_loss_array, c='r',label='test')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(122)
plt.plot(np.arange(len(train_loss_array)) + 1,train_accuracy_array,c='b',label='train')
plt.plot(np.arange(len(train_loss_array)) + 1,val_accuracy_array, c='r',label='test')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```

Out[50]:
```
<matplotlib.legend.Legend at 0x1e15958a110>
```





In [51]:
```
print(f"The Accuracy of the above model on test data is : {max(val_accuracy_array)}")
```

```
The Accuracy of the above model on test data is : 78.42
```

In [52]:
```
print(inference_time)
```

```
(3.7366865622997283, 0.4563889909816836)
```

In [53]:
```
print(flops, params)
```

```
28.406794 11.04353
```

Incremental Replacement of Resenet Layers with MobileNet

In [57]:
```
config = {
    'model': 'resnet18_modified',
    'dataset': 'cifar10',
```

```
    'batch_size': 256,
    'num_workers': 4,
    'lr': 1e-3,
    'epochs': 50,
    'criterion': nn.NLLLoss,
    'optimizer': 'Adam',
    'device': 'cuda' if torch.cuda.is_available() else 'cpu',
    'input_channels': 3,
    'num_classes': 10,
    'pin_memory': True if torch.cuda.is_available() else False,
    'depthwise_convs': [True, False, False, False],
    'lr_scheduler' : "ReduceLROnPlateau"
}

val_loss_arrays = []
val_accuracy_arrays = []
flops_array = []
params_array = []
inference_time_array = []

depthwise_configs = [
    [False, False, False, False], # No blocks repalced
    [True, False, False, False], # Only First
    [True, True, False, False], # First two
    [True, True, True, False],
    [True, True, True, True],
    [False, False, False, True], # Only Last
    [False, False, True, True], # Last two
    [False, True, True, True],
]

for depthwise_config in depthwise_configs:
    config['depthwise_convs'] = depthwise_config
    val_loss_array, val_accuracy_array, train_loss_array, train_accuracy_array, flops, params, inference_time = train_v
        config)

    val_loss_arrays.append(val_loss_array)
    val_accuracy_arrays.append(val_accuracy_array)
    flops_array.append(flops)
    params_array.append(params)
    inference_time_array.append(inference_time)
```

```
Files already downloaded and verified
[INFO]: Epoch 1/50
Train loss: 1.4643, Train accuracy: 46.94
Validation loss: 1.4844, Validation accuracy: 48.28
-------------------------------------------------
[INFO]: Epoch 2/50
Train loss: 1.0573, Train accuracy: 62.14
Validation loss: 1.0570, Validation accuracy: 63.36
-------------------------------------------------
[INFO]: Epoch 3/50
Train loss: 0.8730, Train accuracy: 69.10
Validation loss: 0.8619, Validation accuracy: 70.18
-------------------------------------------------
[INFO]: Epoch 4/50
Train loss: 0.7442, Train accuracy: 73.79
Validation loss: 0.8051, Validation accuracy: 71.99
-------------------------------------------------
[INFO]: Epoch 5/50
Train loss: 0.6628, Train accuracy: 76.55
Validation loss: 0.7561, Validation accuracy: 74.28
-------------------------------------------------
[INFO]: Epoch 6/50
Train loss: 0.5868, Train accuracy: 79.61
Validation loss: 0.7560, Validation accuracy: 74.04
-------------------------------------------------
[INFO]: Epoch 7/50
Train loss: 0.5258, Train accuracy: 81.51
Validation loss: 0.7854, Validation accuracy: 74.19
-------------------------------------------------
[INFO]: Epoch 8/50
Train loss: 0.4617, Train accuracy: 83.79
Validation loss: 0.8098, Validation accuracy: 73.74
-------------------------------------------------
[INFO]: Epoch 9/50
Train loss: 0.4169, Train accuracy: 85.45
Validation loss: 0.7286, Validation accuracy: 76.64
-------------------------------------------------
[INFO]: Epoch 10/50
Train loss: 0.3697, Train accuracy: 86.99
Validation loss: 0.7897, Validation accuracy: 75.53
-------------------------------------------------
[INFO]: Epoch 11/50
Train loss: 0.3328, Train accuracy: 88.27
Validation loss: 0.7733, Validation accuracy: 76.55
-------------------------------------------------
```

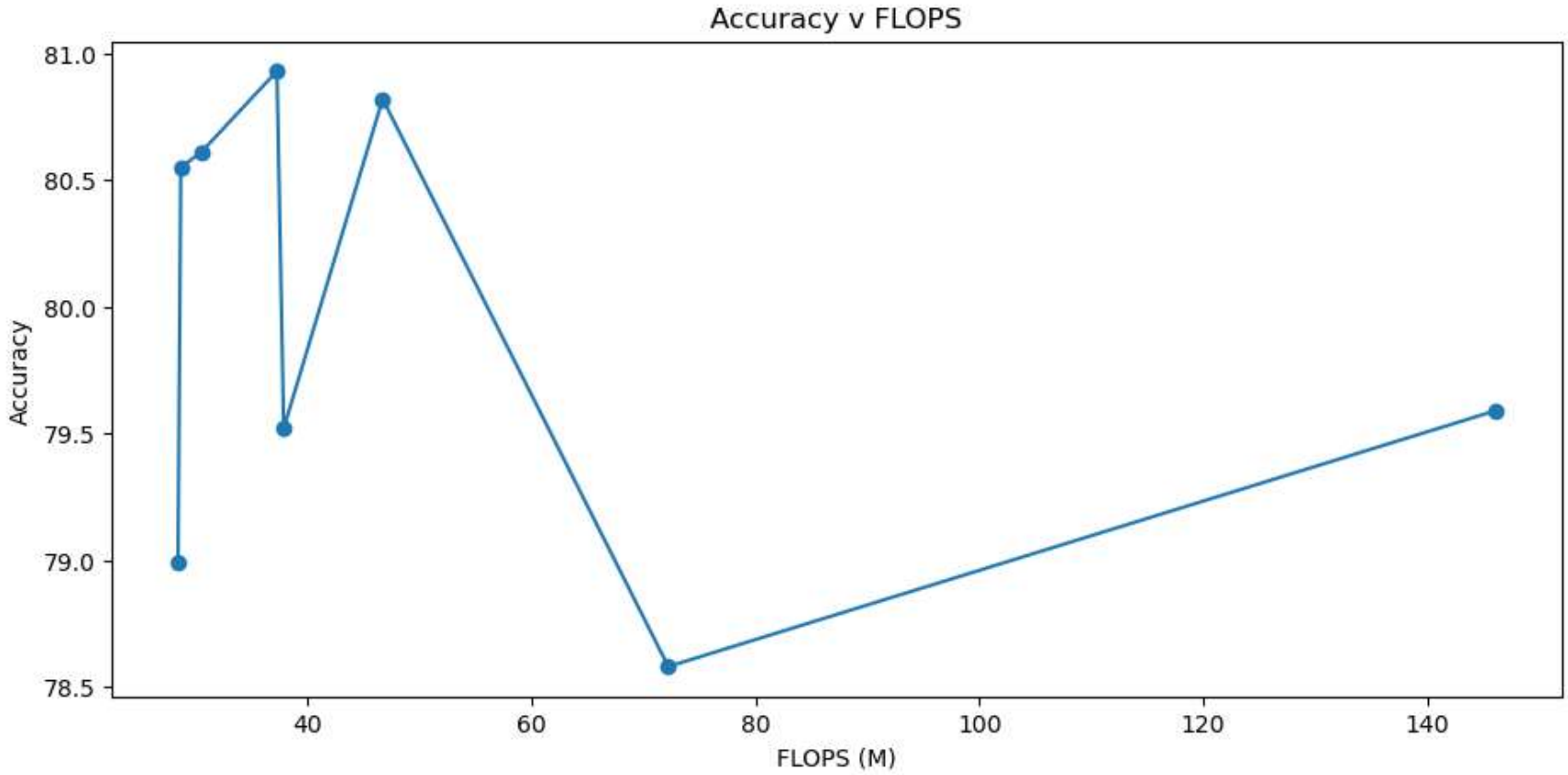
```
Train loss: 0.2914, Train accuracy: 89.87
Validation loss: 0.5892, Validation accuracy: 80.82
-------------------------------------------------
[INFO]: Epoch 24/50
Train loss: 0.2875, Train accuracy: 90.01
Validation loss: 0.5970, Validation accuracy: 80.68
-------------------------------------------------
[INFO]: Epoch 25/50
Train loss: 0.2864, Train accuracy: 89.88
Validation loss: 0.5994, Validation accuracy: 80.63
-------------------------------------------------
Epoch 00025: reducing learning rate of group 0 to 2.5000e-06.
[INFO]: Epoch 26/50
Train loss: 0.2802, Train accuracy: 90.17
Validation loss: 0.5953, Validation accuracy: 80.67
-------------------------------------------------
[INFO]: Epoch 27/50
Train loss: 0.2755, Train accuracy: 90.43
Validation loss: 0.6048, Validation accuracy: 80.58
-------------------------------------------------
[INFO]: Early stopping
```

Efficiency vs Accuracy Tradeoff

In [88]:
```
max_validation_accuracies = [max(i) for i in val_accuracy_arrays]
max_validation_accuracies = np.array(max_validation_accuracies)
```

In [91]:
```
plt.figure(figsize=(11, 5))

flops_array = np.array(flops_array)
ind = np.argsort(flops_array)

plt.plot(flops_array[ind], max_validation_accuracies[ind], marker='o')
plt.title("Accuracy v FLOPS")
plt.xlabel('FLOPS (M)')
plt.ylabel('Accuracy')

list(zip(flops_array,max_validation_accuracies))
```

Out[91]:
```
[(37.245962, 80.93),
 (28.406794, 78.99),
 (72.19713, 78.58),
 (146.097162, 79.59),
 (37.848074, 79.52),
 (30.480394, 80.61),
 (28.658698, 80.55),
 (46.687242, 80.82)]
```

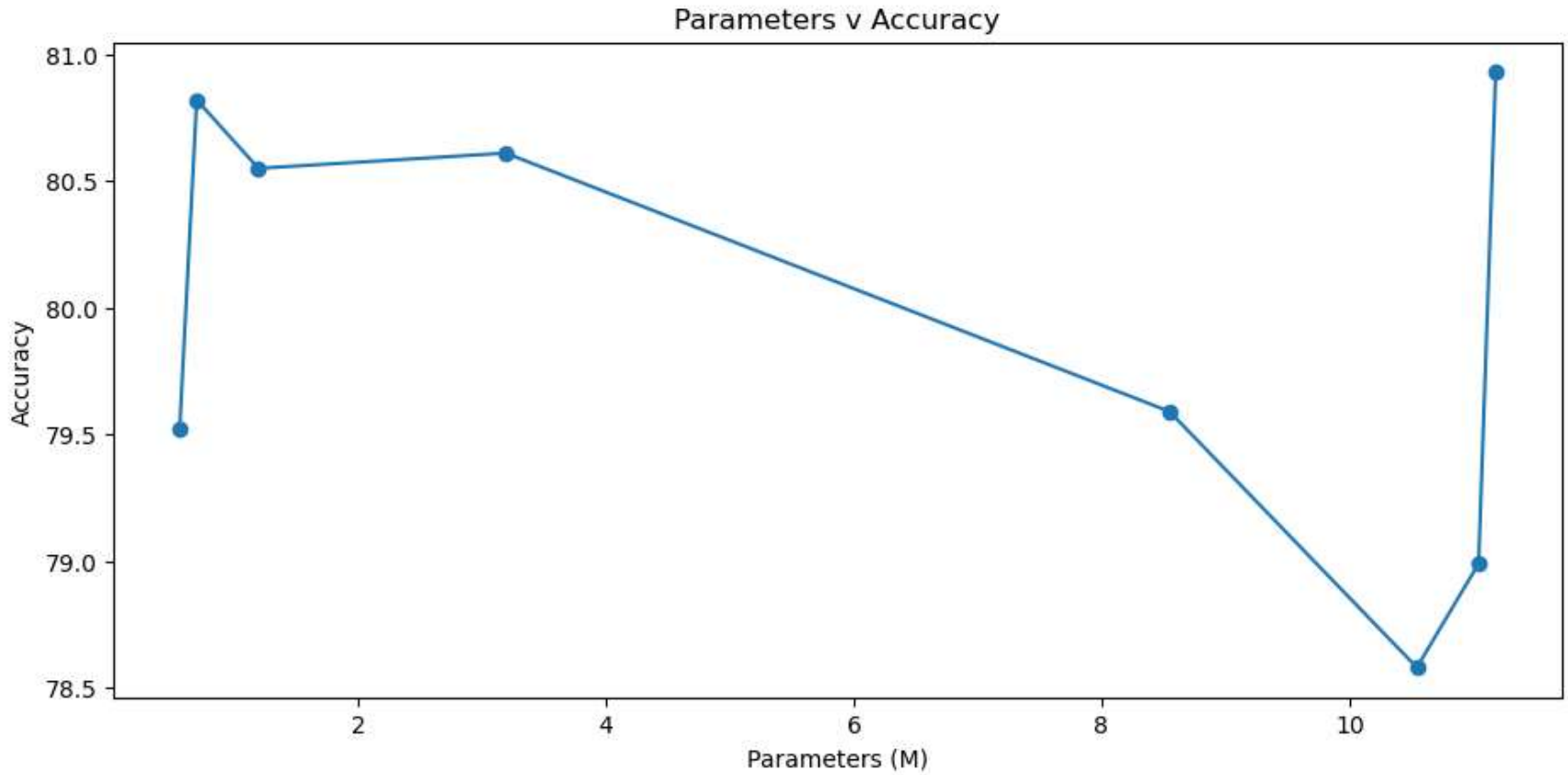


From the above plot we can see that a model with lower FLOPS 46.687M MAC performed the best amongst all other models. This particular model corresponds to the model were the last 3 layers of the ResNet18 were changed to the MobileNet layer. Compared to the best model there was actually an increase in the number of MAC performed with a drop of 0.11% in accuracy.

In [94]:
```
plt.figure(figsize=(11, 5))


params_array = np.array(params_array)
```

```
ind = np.argsort(params_array)

plt.plot(params_array[ind], max_validation_accuracies[ind], marker='o')
plt.title("Parameters v Accuracy")
plt.xlabel('Parameters (M)')
plt.ylabel('Accuracy')

list(zip(params_array,max_validation_accuracies))
```

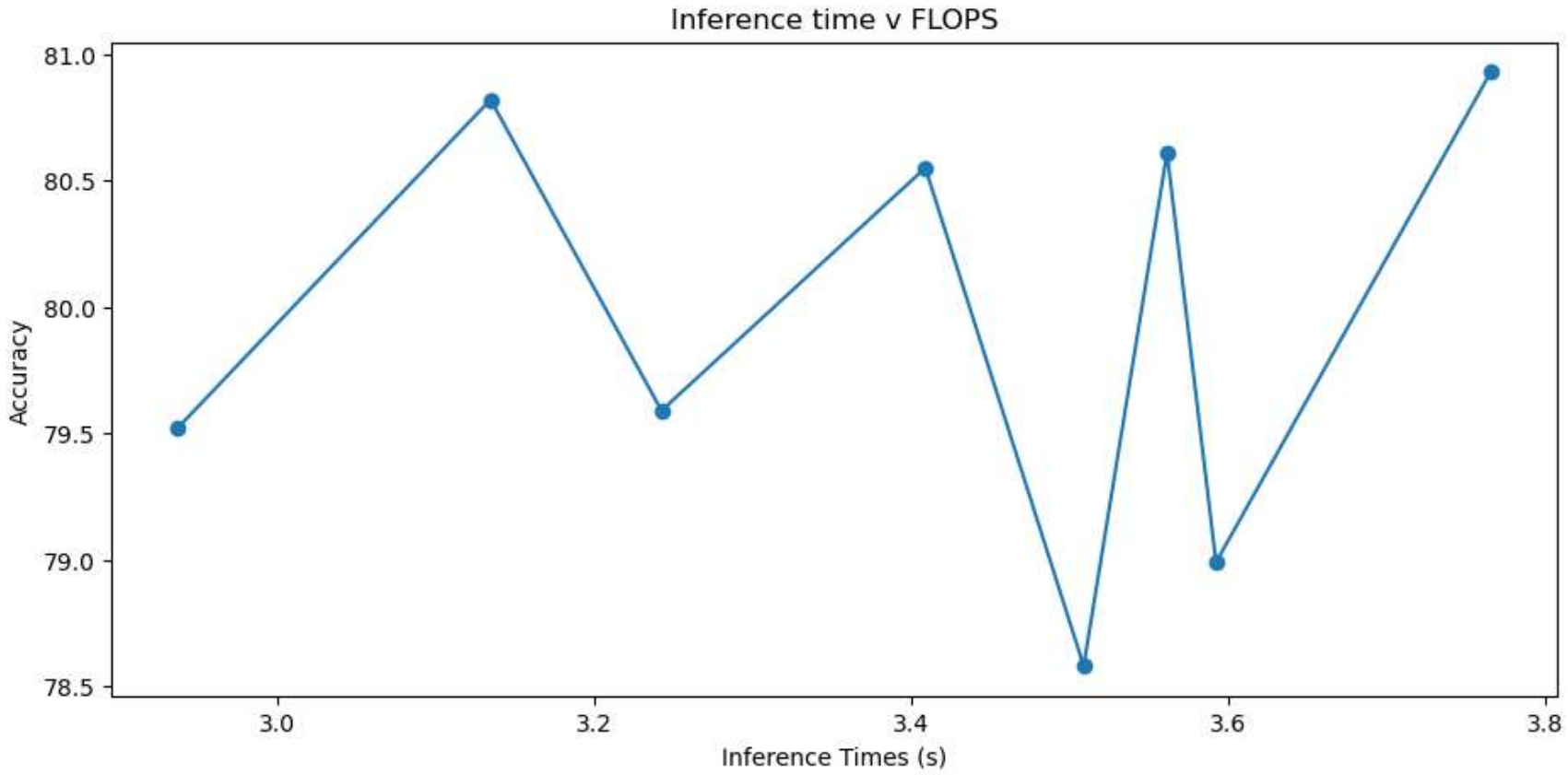
Out[94]:
```
[(11.181642, 80.93),
 (11.04353, 78.99),
 (10.545162, 78.58),
 (8.549002, 79.59),
 (0.558986, 79.52),
 (3.191626, 80.61),
 (1.195466, 80.55),
 (0.697098, 80.82)]
```



From the above plot we can see that the model where the last three ResNet18 Layers were changed to MobileNet layer (0.697 M parameters) performed almost as well as the full ResNet18 model which had 11.18 M parameters. Even the model with only 0.559 M parameters performed simialrly to that of the 11.18M parameter model. The 0.697M parameters model only had a 0.11% reduction in accuracy.

In [99]:
```
plt.figure(figsize=(11, 5))


inference_times = np.array([i[0] for i in inference_time_array])
errors = np.array([i[1] for i in inference_time_array])

ind = np.argsort(inference_times)

plt.errorbar(inference_times[ind], max_validation_accuracies[ind], marker='o')
plt.title("Inference time v FLOPS")
plt.xlabel('Inference Times (s)')
plt.ylabel('Accuracy')

list(zip(inference_times, max_validation_accuracies))
```

Out[99]:
```
[(3.76602672290802, 80.93),
 (3.592393216371536, 78.99),
 (3.5088193876743317, 78.58),
 (3.2422088968753813, 79.59),
 (2.936466114282608, 79.52),
 (3.561467452764511, 80.61),
 (3.4088022706508636, 80.55),
 (3.1342129600048065, 80.82)]
```



We see a similar trend here where, there is a 0.63s increase in inference time, at a loss of 0.11% accuracy. Although this result is not as conclusive as the standard deviations were quite large, this is primarily due to the usage of GPU fluctuating with other processes possible interfering causing instabilities in the calcuation.

Q1

How do you determine the strategy for replacing the building blocks in ResNet-18 with MobileNet Blocks

As there are 4 layers with each layer containing two ResNet blocks, each of these building blocks were replaced incrementally and tested with their performance metric being recorded, i.e. the first block was replaced with the MobileNet block, then the first two and so on. The opposite was also done wherein the last block was replaced then the second last and the last and so on. The results of the test are shown in Section 3.3.

Q2

To what extent can you advance the accuracy-efficiency trade-off, meaning, what is the accuracy drop (or improvement) and what are the efficiency gains?

The expanations for the above is given under the plots of their relevent effciency parameter plots.

Q3

Are the efficiency savings in terms of theoretical metrics proportional to your real-device measurements? Could you analyze why?

In terms of number of parameters used, the efficiency saving leads to a much smaller model, thus the amount of space in memory its occupies is reduced and is proportional to the real-device measurements.

However, metrics such as the inference time, have such a large standard deviation (error-bars), and furthermore are specific to the device it was trained on (My laptop with a RTX 1660). Thus although it is proportional to the savings in my laptop it is not generalizable to other devices.

In [ ]: