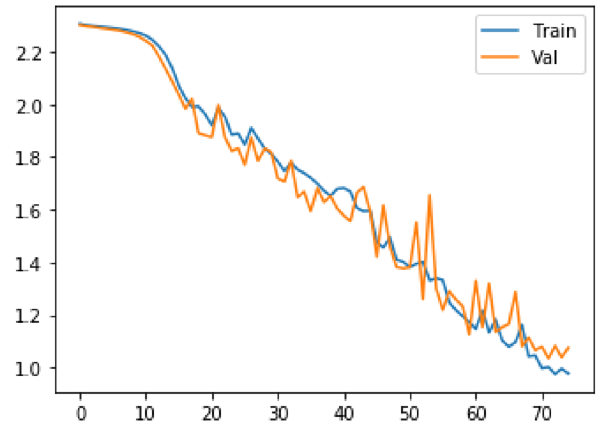
Question 3: RNN_ActionClassify

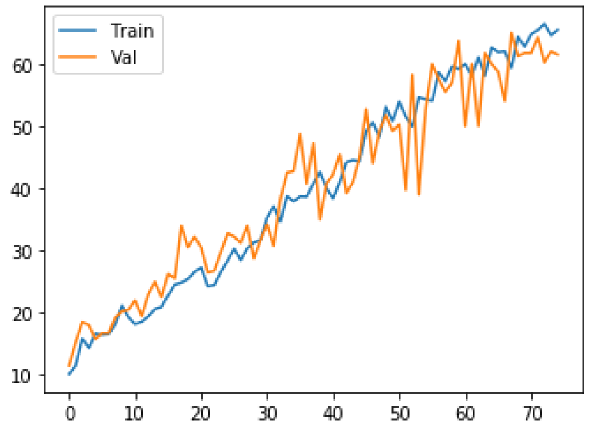
best performance on the Validation set: 87.5

1) baseline: 61.5

```
  (project_layer): Linear(in_features=75, out_features=100, bias=True)
  (recurrent_layer): LSTM(100, 100, batch_first=True)
  (classify_layer): Linear(in_features=100, out_features=10, bias=True)
```

```
optimizer = torch.optim.SGD(model.parameters(), lr=1e-3, momentum=0.9)
criterion = nn.CrossEntropyLoss().type(dtype)
num_epochs = 75
```





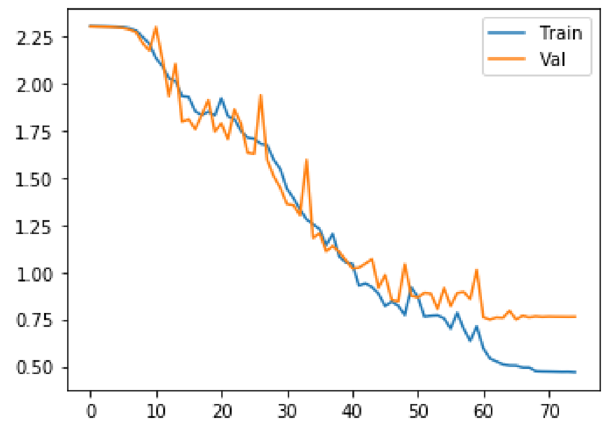
2) 76.25

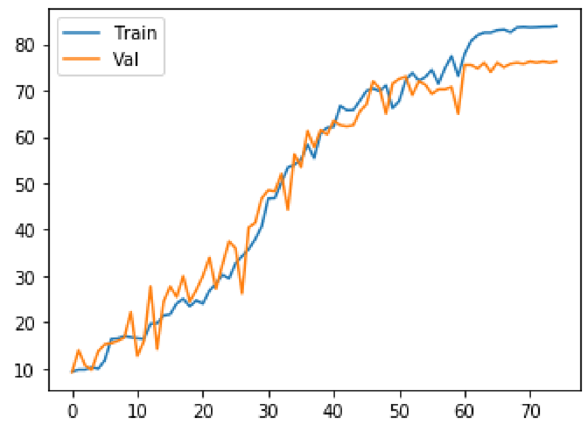
Increased trainable parameters in recurrent and classify layers,

Increased learning rate in SGD optimizer, added LR scheduler

```
(recurrent_layer): LSTM(75, 300, num_layers=3, batch_first=True)
  (classify_layer): Linear(in_features=300, out_features=10, bias=True)
```

```
optimizer = torch.optim.SGD(model.parameters(), lr=1e-2, momentum=0.9)
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, patience=5, factor=0.1)
criterion = nn.CrossEntropyLoss().type(dtype)
num_epochs = 75
```



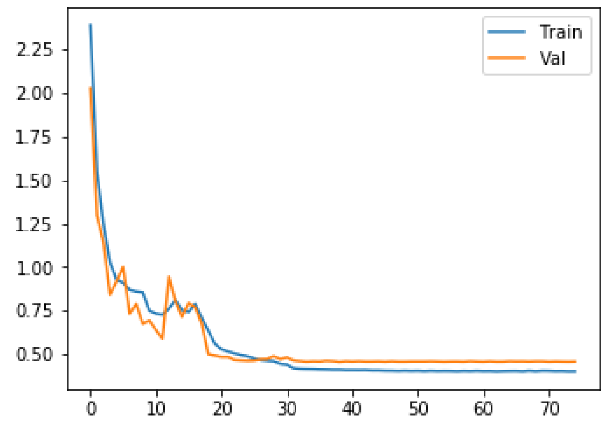


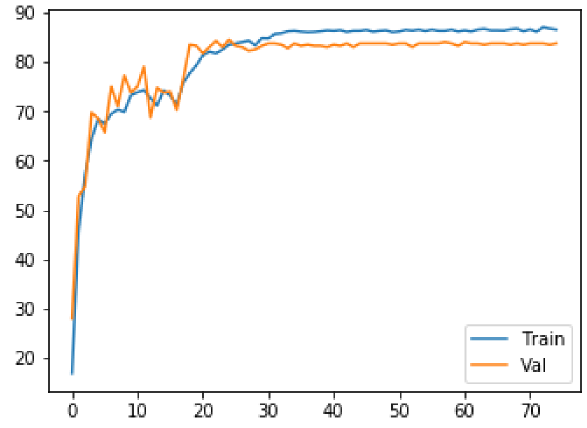
3) 83.75

Reduced trainable parameters in recurrent layer and added conv1d layer in classify layer. Rest same.

```
(recurrent_layer): LSTM(75, 200, batch_first=True)
  (classify_layer): Sequential(
    (0): Conv1d(15, 10, kernel_size=(3,), stride=(1,))
    (1): BatchNorm1d(10, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Flatten()
    (4): Linear(in_features=1980, out_features=10, bias=True)
  )
```

```
optimizer = torch.optim.SGD(model.parameters(), lr=1e-2, momentum=0.9)
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, patience=5, factor=0.1)
criterion = nn.CrossEntropyLoss().type(dtype)
num_epochs = 75
```



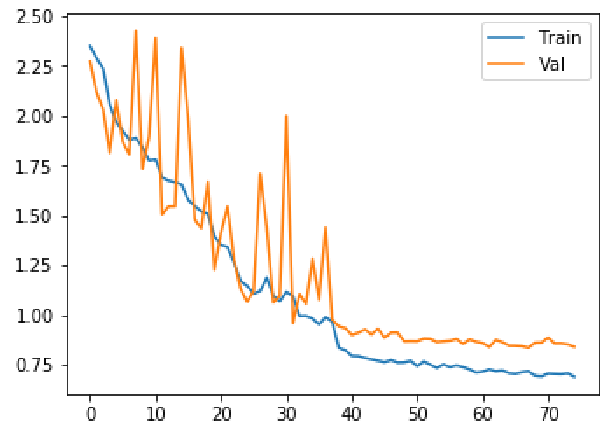


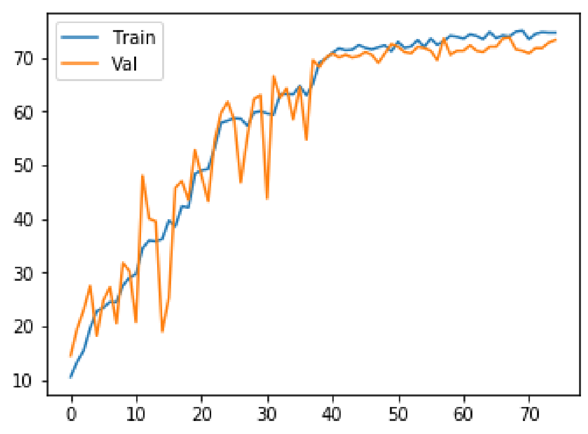
4) 73.25

changed recurrent layer with dropout, bidirectional addition. Added one more lstm layer

```
(recurrent_layer): LSTM(75, 200, num_layers=2, batch_first=True, dropout=0.8,
bidirectional=True)
  (r2): LSTM(75, 50, num_layers=2, batch_first=True, dropout=0.8)
  (classify_layer): Sequential(
    (0): Linear(in_features=450, out_features=128, bias=True)
    (1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
    (3): Linear(in_features=128, out_features=10, bias=True)
  )
```

```
optimizer = torch.optim.SGD(model.parameters(), lr=1e-2, momentum=0.9)
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, patience=5, factor=0.1)
criterion = nn.CrossEntropyLoss().type(dtype)
num_epochs = 75
```



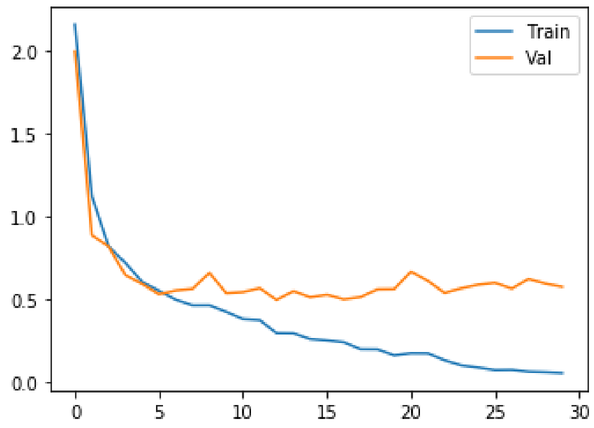


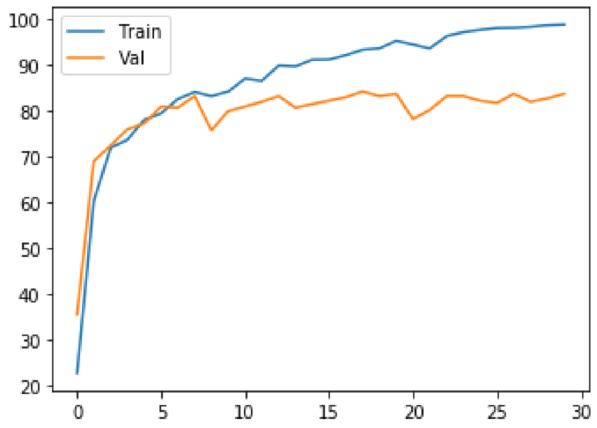
5) 83.75

From 3rd attempt, increased 1 layer in LSTM, used LeakyReLU in classify layer, used Adam optimizer

```
(recurrent_layer): LSTM(75, 200, num_layers=2, batch_first=True)
  (classify_layer): Sequential(
    (0): Conv1d(15, 10, kernel_size=(3,), stride=(1,))
    (1): BatchNorm1d(10, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
    (3): Flatten()
    (4): Linear(in_features=1980, out_features=10, bias=True)
  )
```

```
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)
criterion = nn.CrossEntropyLoss().type(dtype)
num_epochs = 30
```



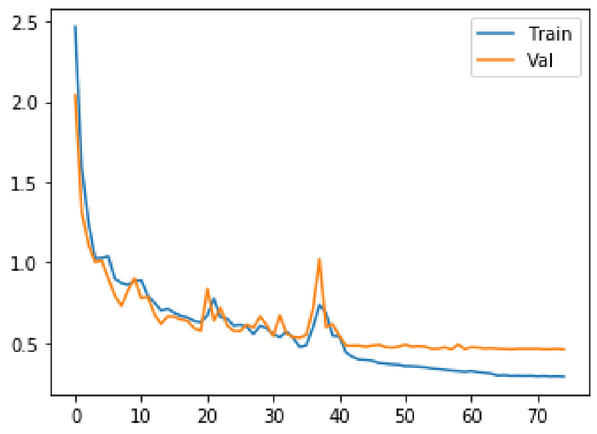


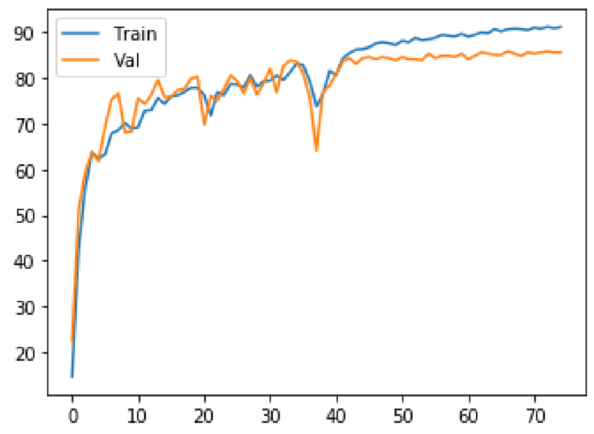
6) 85.5

From last attempt, again kept 1 layer in recurrent layer, used SGD optimizer (both same as 3rd attempt)

```
(recurrent_layer): LSTM(75, 200, batch_first=True)
  (classify_layer): Sequential(
    (0): Conv1d(15, 10, kernel_size=(3,), stride=(1,))
    (1): BatchNorm1d(10, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
    (3): Flatten()
    (4): Linear(in_features=1980, out_features=10, bias=True)
  )
```

```
optimizer = torch.optim.SGD(model.parameters(), lr=1e-2, momentum=0.9)
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, patience=5, factor=0.1)
criterion = nn.CrossEntropyLoss().type(dtype)
num_epochs = 75
```





7) 87.5

Changed learning rate = 0.7*1e-3, added weight decay = 0.02

```
(recurrent_layer): LSTM(75, 200, batch_first=True)
  (classify_layer): Sequential(
    (0): Conv1d(15, 10, kernel_size=(3,), stride=(1,))
    (1): BatchNorm1d(10, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
    (3): Flatten()
    (4): Linear(in_features=1980, out_features=10, bias=True)
  )
)
```

```
optimizer = torch.optim.SGD(model.parameters(), lr=0.7*1e-
3, momentum=0.9, weight_decay=0.02)
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, patience=3, factor=0.1, ve
rbose=True, min_lr=5e-5)
criterion = nn.CrossEntropyLoss().type(dtype)
num_epochs = 100
```