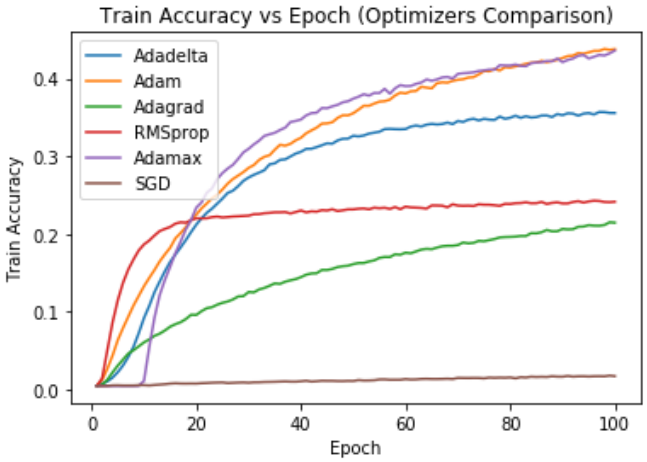
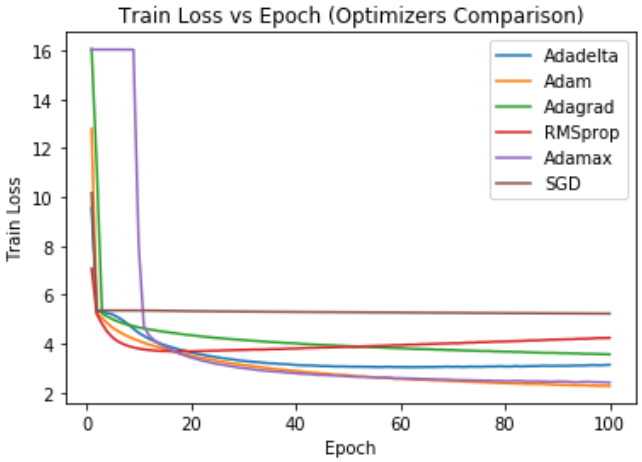
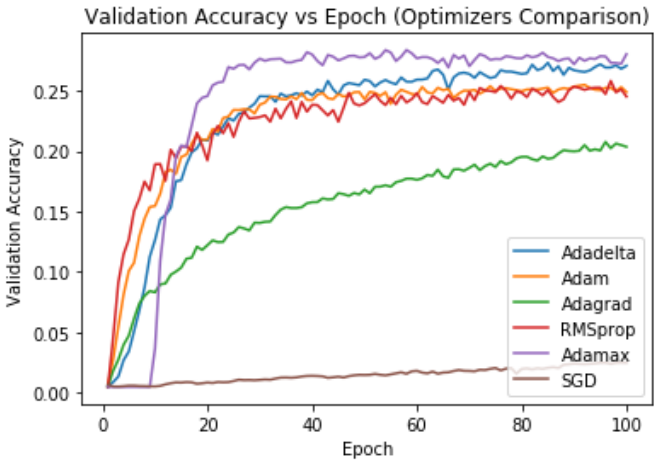
In [1]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

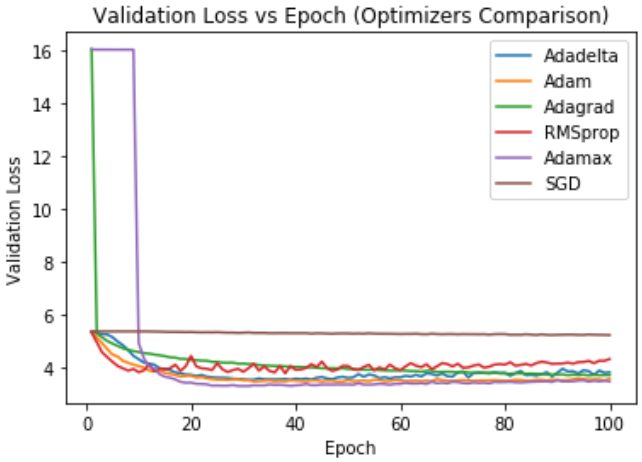
In [2]:
```
def getAcc_Loss(path, epoch):
    file = open(path, "r")

    # this will store the train and validation accuracy and loss
    train_acc = []
    train_loss = []
    val_acc = []
    val_loss = []

    # get the result for plotting
    for i, line in enumerate(file):
        if i < 2:
            continue
        elif i < 2 + epoch:
            result = [float(r) for r in line.split("$")]
            train_acc.append(result[0])
            train_loss.append(result[1])
        elif i == 2 + epoch:
            continue
        elif i == epoch * 2 + 3:
            continue
        else:
            result = [float(r) for r in line.split("$")]
            val_acc.append(result[0])
            val_loss.append(result[1])

    file.close()

    return (train_acc, train_loss, val_acc, val_loss)
```

In [3]:
```
# plot for VGG-like
epoch = 100
yAxis = range(1, epoch+1)

# get the results
adadeltaResult = getAcc_Loss("./Results/VGG-like Adadelta.txt",  epoch)
adamResult = getAcc_Loss("./Results/VGG-like Adam.txt",  epoch)
adagradResult = getAcc_Loss("./Results/VGG-like Adagrad.txt",  epoch)
RMSpropResult = getAcc_Loss("./Results/VGG-like RMSprop.txt",  epoch)
adamaxResult = getAcc_Loss("./Results/VGG-like Adamax.txt",  epoch)
sgdResult = getAcc_Loss("./Results/VGG-like SGD.txt",  epoch)

for i in range(4):
    plt.plot(yAxis, adadeltaResult[i], label = 'Adadelta')
    plt.plot(yAxis, adamResult[i], label = 'Adam')
    plt.plot(yAxis, adagradResult[i], label = 'Adagrad')
    plt.plot(yAxis, RMSpropResult[i], label = 'RMSprop')
    plt.plot(yAxis, adamaxResult[i], label = 'Adamax')
    plt.plot(yAxis, sgdResult[i], label = 'SGD')

    yAxisLabel = {
        0: "Train Accuracy",
        1: "Train Loss",
        2: "Validation Accuracy",
        3: "Validation Loss"
    }

    plt.xlabel('Epoch')
    myTitle = yAxisLabel[i]
    plt.ylabel(myTitle)
    plt.title("%s vs Epoch (Optimizers Comparison)" % (myTitle))
    plt.legend(loc='best')
    plt.savefig("./Plots/%s vs Epoch.png" % (myTitle))
    plt.show()
```

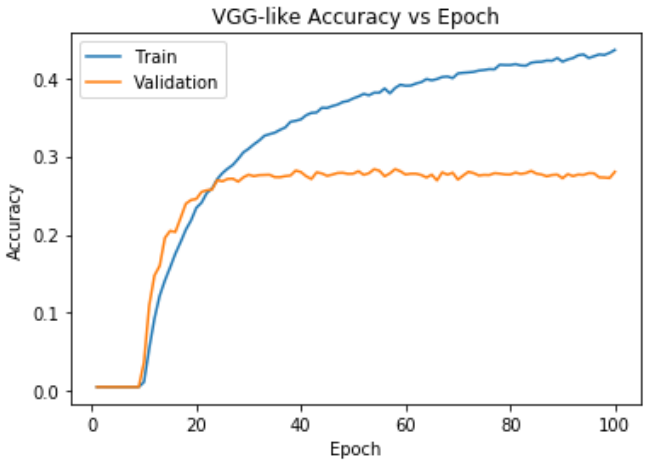

Train Accuracy vs Epoch (Optimizers Comparison)
Adadelta
Adam
Adagrad
RMSprop
Adamax
SGD
Train Accuracy
Epoch


Train Loss vs Epoch (Optimizers Comparison)
Adadelta
Adam
Adagrad
RMSprop
Adamax
SGD
Train Loss
Epoch

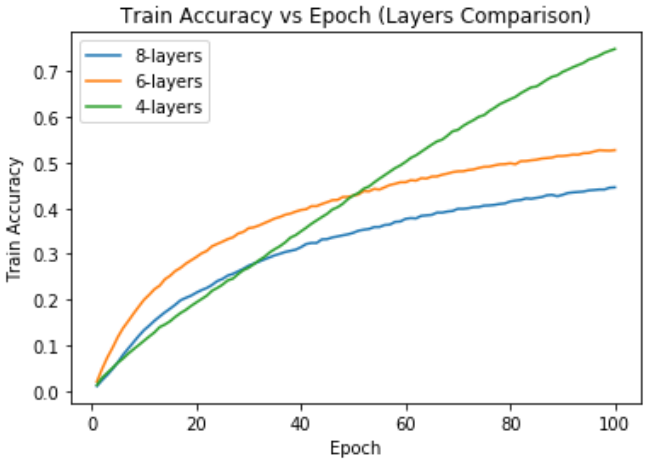

Validation Accuracy vs Epoch (Optimizers Comparison)
Adadelta
Adam
Adagrad
RMSprop
Adamax
SGD
Validation Accuracy
Epoch

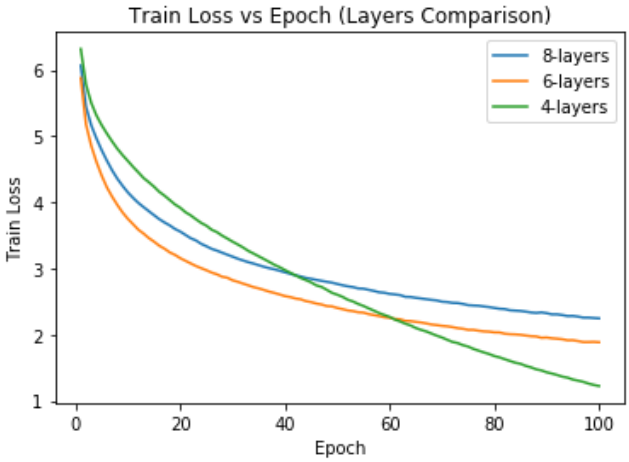


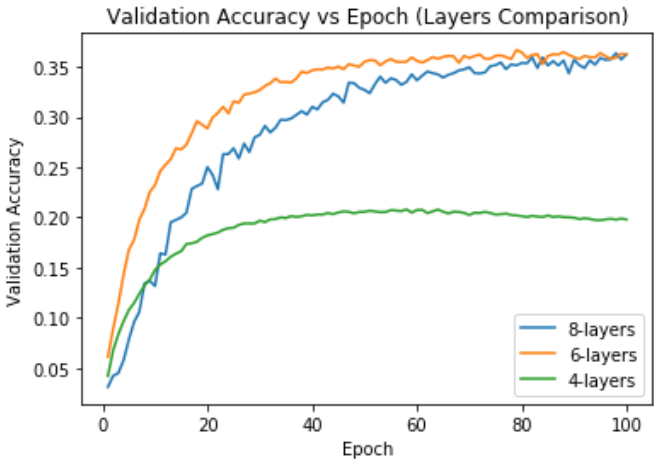
In [4]:
```
# plot best accuracy for VGG-like
plt.plot(yAxis, adamaxResult[0], label = 'Train')
plt.plot(yAxis, adamaxResult[2], label = 'Validation')

plt.xlabel('Epoch')
plt.ylabel("Accuracy")
plt.title("VGG-like Accuracy vs Epoch")
plt.legend(loc='best')
plt.savefig("./Plots/VGG-like Accuracy vs Epoch.png")
plt.show()
```

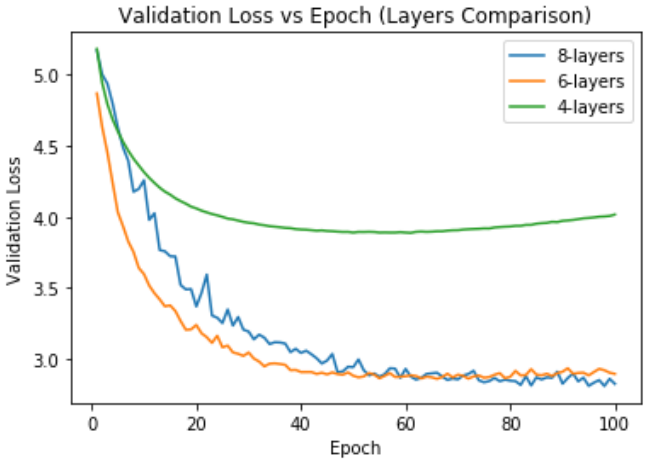


In [5]:
```
# plot for layers
epoch = 100
yAxis = range(1, epoch+1)

# get the results
model1Result = getAcc_Loss("./Results/model1.txt",  epoch)
model2Result = getAcc_Loss("./Results/model2.txt",  epoch)
model3Result = getAcc_Loss("./Results/model3.txt",  epoch)

for i in range(4):
    plt.plot(yAxis, model1Result[i], label = '8-layers')
    plt.plot(yAxis, model2Result[i], label = '6-layers')
    plt.plot(yAxis, model3Result[i], label = '4-layers')

    yAxisLabel = {
        0: "Train Accuracy",
        1: "Train Loss",
        2: "Validation Accuracy",
        3: "Validation Loss"
    }

    plt.xlabel('Epoch')
    myTitle = yAxisLabel[i]
    plt.ylabel(myTitle)
    plt.title("%s vs Epoch (Layers Comparison)" % (myTitle))
    plt.legend(loc='best')
    plt.savefig("./Plots/Layers %s vs Epoch.png" % (myTitle))
    plt.show()
```

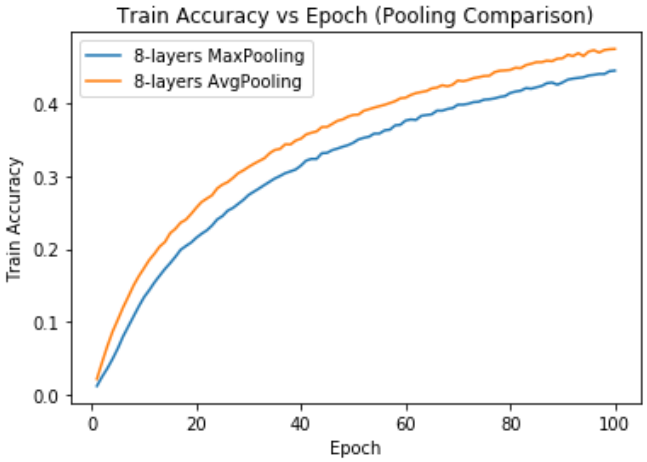

Train Accuracy vs Epoch (Layers Comparison)
8-layers
6-layers
4-layers
Train Accuracy
Epoch

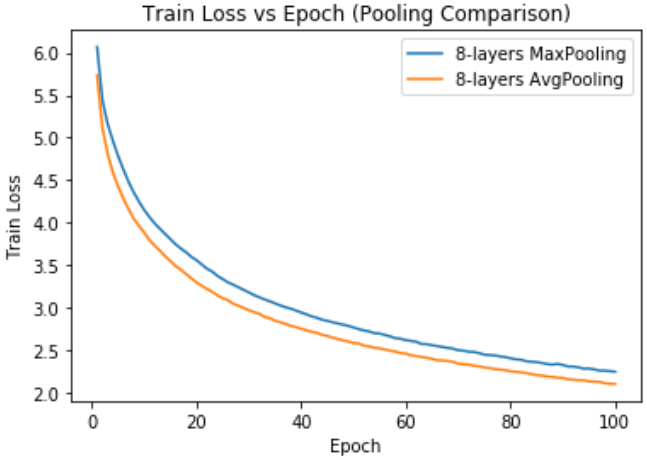

Train Loss vs Epoch (Layers Comparison)
8-layers
6-layers
4-layers
Train Loss
Epoch

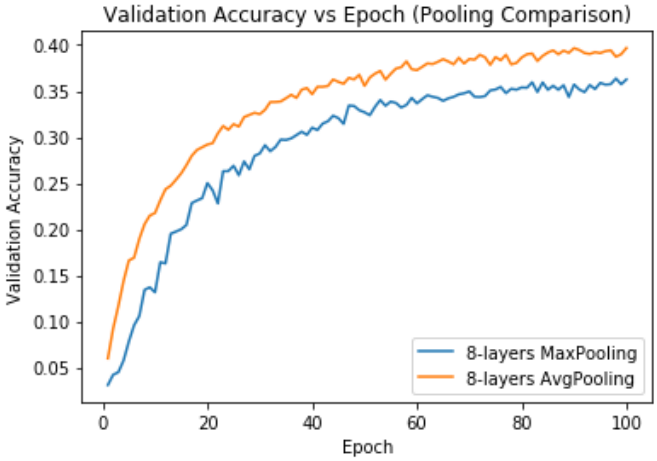

Validation Accuracy vs Epoch (Layers Comparison)
8-layers
6-layers
4-layers
Validation Accuracy
Epoch

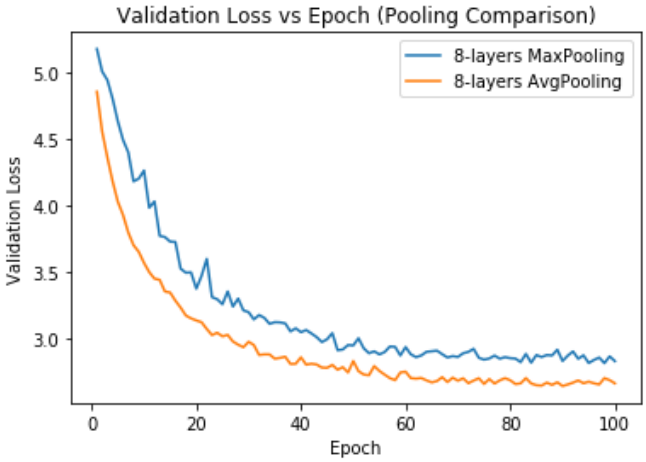

Validation Loss vs Epoch (Layers Comparison)
8-layers
6-layers
4-layers
Validation Loss
5.0
4.5
4.0
3.5
3.0
0
20
40
60
80
100
Epoch

In [6]:
```
# plot for poolings
epoch = 100
yAxis = range(1, epoch+1)

# get the results
model1LuResult = getAcc_Loss("./Results/model1.txt",  epoch)
model1AvgResult = getAcc_Loss("./Results/model1_avg.txt",  epoch)

for i in range(4):
    plt.plot(yAxis, model1Result[i], label = '8-layers MaxPooling')
    plt.plot(yAxis, model1AvgResult[i], label = '8-layers AvgPooling')

    yAxisLabel = {
        0: "Train Accuracy",
        1: "Train Loss",
        2: "Validation Accuracy",
        3: "Validation Loss"
    }

    plt.xlabel('Epoch')
    myTitle = yAxisLabel[i]
    plt.ylabel(myTitle)
    plt.title("%s vs Epoch (Pooling Comparison)" % (myTitle))
    plt.legend(loc='best')
    plt.savefig("./Plots/Pools %s vs Epoch.png" % (myTitle))
    plt.show()
```

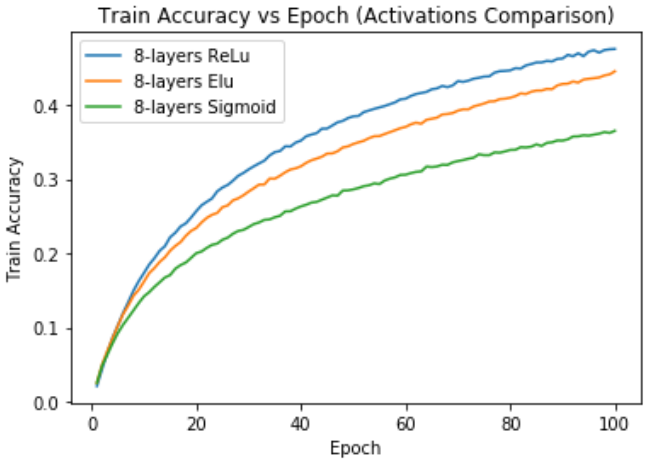

Train Accuracy vs Epoch (Pooling Comparison)
8-layers MaxPooling
8-layers AvgPooling
Train Accuracy
Epoch

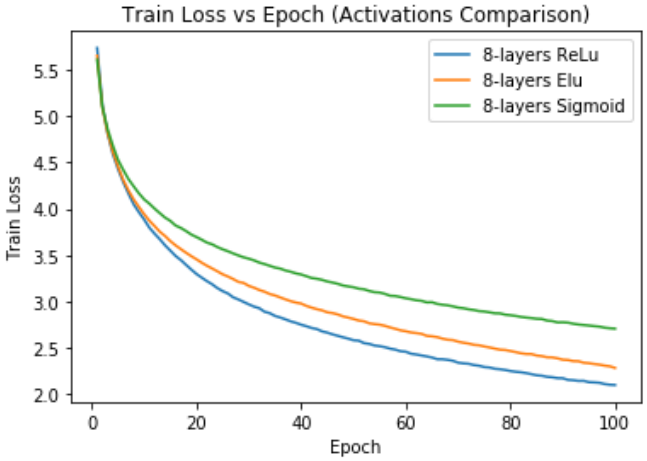

Train Loss vs Epoch (Pooling Comparison)
8-layers MaxPooling
8-layers AvgPooling
Train Loss
Epoch

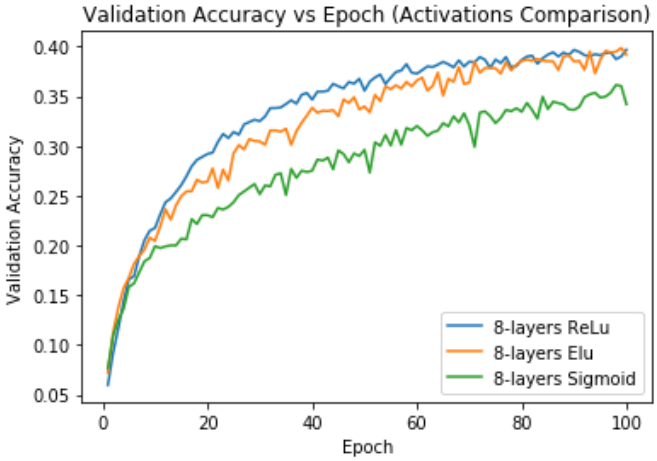

Validation Accuracy vs Epoch (Pooling Comparison)
8-layers MaxPooling
8-layers AvgPooling
Validation Accuracy
Epoch

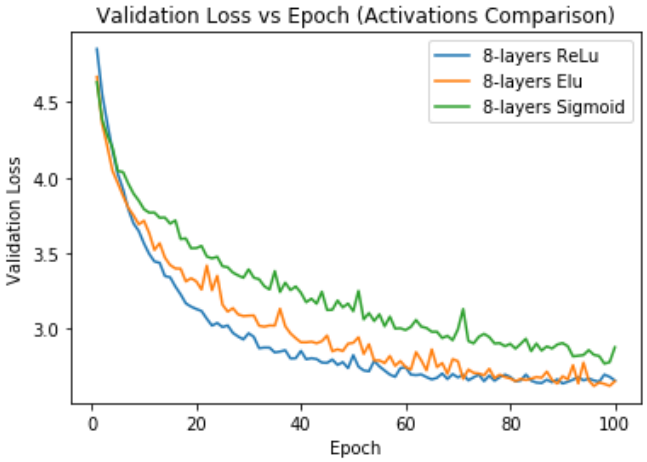

Validation Loss vs Epoch (Pooling Comparison)
8-layers MaxPooling
8-layers AvgPooling
Validation Loss
5.0
4.5
4.0
3.5
3.0
0
20
40
60
80
100
Epoch

In [7]:
```
# plot for activations
epoch = 100
yAxis = range(1, epoch+1)

# get the results
model1Result = getAcc_Loss("./Results/model1_avg.txt",  epoch)
model1EluResult = getAcc_Loss("./Results/model1_elu.txt",  epoch)
model1SigmoidResult = getAcc_Loss("./Results/model1_sigmoid.txt",  epoch)

for i in range(4):
    plt.plot(yAxis, model1Result[i], label = '8-layers ReLu')
    plt.plot(yAxis, model1EluResult[i], label = '8-layers Elu')
    plt.plot(yAxis, model1SigmoidResult[i], label = '8-layers Sigmoid')

    yAxisLabel = {
        0: "Train Accuracy",
        1: "Train Loss",
        2: "Validation Accuracy",
        3: "Validation Loss"
    }

    plt.xlabel('Epoch')
    myTitle = yAxisLabel[i]
    plt.ylabel(myTitle)
    plt.title("%s vs Epoch (Activations Comparison)" % (myTitle))
    plt.legend(loc='best')
    plt.savefig("./Plots/Activations %s vs Epoch.png" % (myTitle))
    plt.show()
```

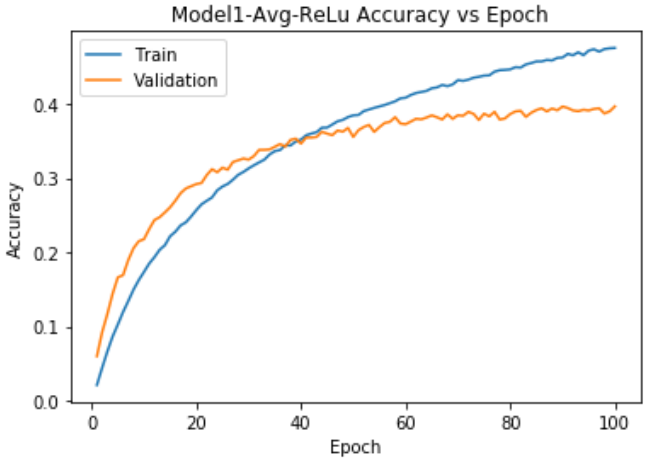

Train Accuracy vs Epoch (Activations Comparison)
8-layers ReLu
8-layers Elu
8-layers Sigmoid
Train Accuracy
Epoch


Train Loss vs Epoch (Activations Comparison)
8-layers ReLu
8-layers Elu
8-layers Sigmoid
Train Loss
Epoch


Validation Accuracy vs Epoch (Activations Comparison)
8-layers ReLu
8-layers Elu
8-layers Sigmoid
Validation Accuracy
Epoch



In [8]:
```
# plot best accuracy for our Model
plt.plot(yAxis, model1Result[0], label = 'Train')
plt.plot(yAxis, model1Result[2], label = 'Validation')

plt.xlabel('Epoch')
plt.ylabel("Accuracy")
plt.title(" Model1-Avg-ReLu Accuracy vs Epoch")
plt.legend(loc='best')
plt.savefig("./Plots/Model1-Avg-ReLu Accuracy vs Epoch.png")
plt.show()
```



In [9]:
```
# plot for activations
epoch = 100
yAxis = range(1, epoch+1)

# get the results
VGG16Result = getAcc_Loss("./Results/VGG-16.txt",  epoch)

plt.plot(yAxis, VGG16Result[0], label = 'Train')
plt.plot(yAxis, VGG16Result[2], label = 'Validation')

plt.xlabel('Epoch')
plt.ylabel("Accuracy")
plt.title("VGG-16 Accuracy vs Epoch")
plt.legend(loc='best')
plt.savefig("./Plots/VGG-16 Accuracy vs Epoch.png")
plt.show()

plt.plot(yAxis, VGG16Result[1], label = 'Train')
plt.plot(yAxis, VGG16Result[3], label = 'Validation')

plt.xlabel('Epoch')
plt.ylabel("Loss")
plt.title("VGG-16 Loss vs Epoch")
plt.legend(loc='best')
plt.savefig("./Plots/VGG-16 Loss vs Epoch.png")
plt.show()
```