In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.nn.init as init
import os
from datetime import datetime

from torch.utils.data import Dataset, DataLoader
```

Question 2

CNN training: Train a Convolutional Neural Network on the simulated data for each of the nine simulation settings. The goal is to use the CNN to predict the cancer status yi based on the simulated images Xi . Additionally, generate a test set of 1000 subjects using the same data generation process and evaluate the CNN's performance in terms of classification accuracy. You are free to build a CNN with arbitrary hyperparameter setting. Conduct at least 10 independent experiments for each setting by generating new datasets each time, and report the hyperparameters for the CNN, the mean and standard deviation of the classification accuracy achieved by your CNN model.

MODEL 2

In [3]:
```
model2 = torch.nn.Sequential()
model2.add_module('conv1', torch.nn.Conv2d(in_channels=1, out_channels=2, kernel_size
model2.add_module('relu1', torch.nn.ReLU())
model2.add_module('pool1', torch.nn.MaxPool2d(kernel_size = 2))


model2.add_module('conv2', torch.nn.Conv2d(in_channels=2, out_channels=4, kernel_size
model2.add_module('relu2', torch.nn.ReLU())
model2.add_module('pool2', torch.nn.MaxPool2d(kernel_size = 2))


model2.add_module('conv3', torch.nn.Conv2d(in_channels=4, out_channels=8, kernel_size
model2.add_module('relu3', torch.nn.ReLU())
model2.add_module('pool3', torch.nn.MaxPool2d(kernel_size = 2))


model2.add_module('Flatten', torch.nn.Flatten())

model2.add_module('fc1', torch.nn.Linear(128, 10))
model2.add_module('relu7', torch.nn.ReLU())
model2.add_module('fc2', torch.nn.Linear(10, 1))

model2.add_module('sigmoid', torch.nn.Sigmoid())
```

In [4]:
```
def simulateData(n, mu_c, mu_n):

    y = np.random.choice([0, 1], size = n, p = [0.5, 0.5])
    m_i = np.random.poisson(lam = mu_c, size = n) * y + np.random.poisson(lam = mu_n,
```

```
    simulated_data = np.zeros([n, 32, 32])
    for i in range(n):
        random_indices = np.random.choice(32 * 32, m_i[i], replace = False)
        row_indices, col_indices = np.unravel_index(random_indices, (32, 32))
        Bi = np.zeros([32, 32])
        Bi[row_indices, col_indices] = 1
        epsilon_i = np.random.normal(loc = 0, scale = np.sqrt(0.04), size = (32, 32))
        simulated_data[i] = Bi + epsilon_i

    return y, simulated_data

class dataSetPytorch(Dataset):
    def __init__(self, x, y):
        self.x = torch.from_numpy(x.reshape([-1, 1, 32, 32])).float()
        self.y = torch.from_numpy(y)
    def __len__(self):
        return len(self.x)
    def __getitem__(self, idx):
        return self.x[idx], self.y[idx]

# Use that to make train and validation data here.
def makeDataLoader(numExperiments = 10):
    n = [200, 500, 1000, 200, 500, 1000, 200, 500, 1000]
    mu_n = [5, 5, 5, 5, 5, 5, 5, 5, 5]
    mu_c = [10, 10, 10, 20, 20, 20, 30, 30, 30]

    dataLoader_experiment_data = []
    for experiment in range(numExperiments):
        dataLoader_settings = []
        for setting in range(9):

            y, simulated_data = simulateData(n = n[setting],
                                             mu_c = mu_c[setting],
                                             mu_n = mu_n[setting])

            datasetSetting = dataSetPytorch(simulated_data, y)
            dataLoader = DataLoader(datasetSetting, batch_size=25, shuffle = True)
            dataLoader_settings.append(dataLoader)

        dataLoader_experiment_data.append(dataLoader_settings)

    return dataLoader_experiment_data

def makeTestLoader(numExperiments = 10):
    n_test = 1000
    mu_n = [5, 5, 5, 5, 5, 5, 5, 5, 5]
    mu_c = [10, 10, 10, 20, 20, 20, 30, 30, 30]

    dataLoader_experiment_data = []
    for experiment in range(numExperiments):
        dataLoader_settings = []
        for setting in range(9):

            y, simulated_data = simulateData(n = n_test,
                                             mu_c = mu_c[setting],
                                             mu_n = mu_n[setting])

            datasetSetting = dataSetPytorch(simulated_data, y)
            dataLoader = DataLoader(datasetSetting, batch_size=25, shuffle = True)
            dataLoader_settings.append(dataLoader)
```

```
        dataLoader_experiment_data.append(dataLoader_settings)

    return dataLoader_experiment_data
```

In [5]:
```
dataLoader_all_experiments_train = makeDataLoader(numExperiments = 10)
dataLoader_all_experiments_val = makeDataLoader(numExperiments = 10)
```

Training the models

In [6]:
```
def reset_weights(model):
    for m in model.modules():
        if isinstance(m, nn.Conv2d) or isinstance(m, nn.Linear):
            init.xavier_uniform_(m.weight)
    return

def saveModel(model, path):
    torch.save(model.state_dict(), path)
```

In [7]:
```
def train(name, model, train_dl, valid_dl, num_epochs = 200):
    # reinitialize weights!
    reset_weights(model)

    loss_fn = torch.nn.BCELoss()
    optimizer = torch.optim.Adam(model.parameters(), lr = 0.001)
    loss_hist_train = [0] * num_epochs
    accuracy_hist_train = [0] * num_epochs
    loss_hist_valid = [0] * num_epochs
    accuracy_hist_valid = [0] * num_epochs

    best_loss = torch.inf

    for epoch in range(num_epochs):
        model.train()

        for x_batch, y_batch in train_dl:
            pred = model(x_batch)[:, 0]
            loss = loss_fn(pred, y_batch.float())
#             print("pred", pred, "observed", y_batch)
            loss.backward()
            optimizer.step()
            optimizer.zero_grad()
            loss_hist_train[epoch] += loss.item() * y_batch.size(0)

            is_correct = ((pred >= 0.5).float() == y_batch).float()
#             print(is_correct)
            accuracy_hist_train[epoch] += is_correct.sum()
        loss_hist_train[epoch] /= len(train_dl.dataset)
        accuracy_hist_train[epoch] /= len(train_dl.dataset)

        model.eval()
        with torch.no_grad():
            for x_batch, y_batch in valid_dl:
                pred = model(x_batch)[:, 0]
                loss = loss_fn(pred, y_batch.float())
                loss_hist_valid[epoch] += loss.item() * y_batch.size(0)
```

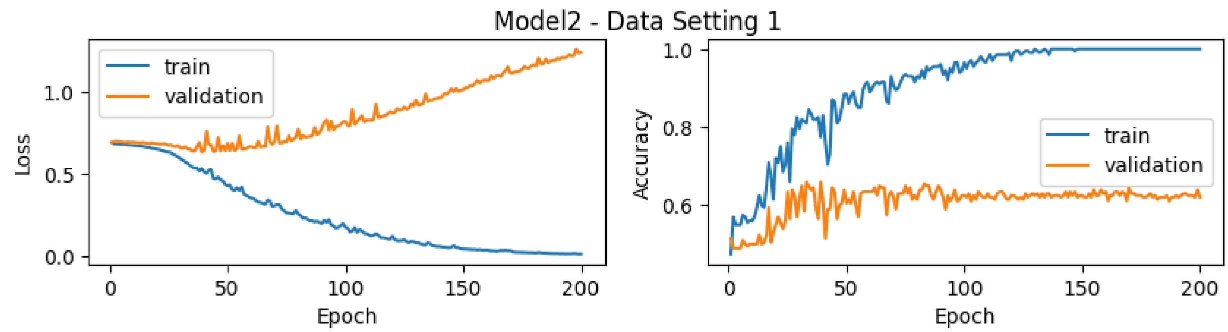
```
                is_correct = ((pred >= 0.5).float() == y_batch).float()
                accuracy_hist_valid[epoch] += is_correct.sum()
        loss_hist_valid[epoch] /= len(valid_dl.dataset)
        accuracy_hist_valid[epoch] /= len(valid_dl.dataset)

        if (best_loss > loss_hist_valid[epoch]):
            path = name + "_" + datetime.now().strftime("%d_%m_%Y") + "_epoch_" + str(
            saveModel(model, path)
            best_loss = loss_hist_valid[epoch]

    return loss_hist_train, loss_hist_valid, accuracy_hist_train, accuracy_hist_valid


def plotModelPerformance(modelNum, dataSettingNum, loss_hist_train, loss_hist_valid,
                         accuracy_hist_train, accuracy_hist_valid):
    epochsX = np.arange(len(loss_hist_train)) + 1
    plt.figure(figsize=(10, 2))
    plt.subplot(1, 2, 1)
    plt.plot(epochsX, loss_hist_train, label = "train")
    plt.plot(epochsX, loss_hist_valid, label = "validation")

    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend()

    plt.subplot(1, 2, 2)
    epochsX = np.arange(len(accuracy_hist_train)) + 1
    plt.plot(epochsX, accuracy_hist_train, label = "train")
    plt.plot(epochsX, accuracy_hist_valid, label = "validation")
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend()

    plt.suptitle("Model" + str(modelNum) + " - Data Setting " + str(dataSettingNum))
    plt.show()
    return
```

Note: In the print statment its supposed to be print("Experiment:", experiment, "Setting:", setting + 1) but forgot to change it when i ran the long experiment ..

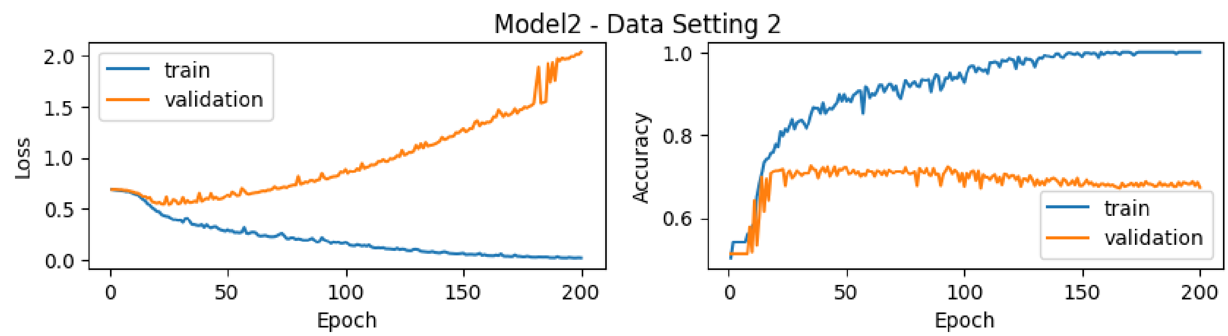
In [9]:
```
numExperiments = 10
numSettings = 9
for experiment in range(numExperiments):
    dataLoader_individual_experiment_train = dataLoader_all_experiments_train[experime
    dataLoader_individual_experiment_val = dataLoader_all_experiments_val[experiment]

    for setting in range(numSettings):
        print("Experiment:", experiment, "Setting:", setting)
        folderPath = 'setting' + str(setting + 1)
        if not os.path.exists(folderPath):
            # Create the folder
            os.makedirs(folderPath)

        n = len(dataLoader_individual_experiment_train[setting].dataset)
        print("N:", n)

        modelName = "./" + folderPath + "/modelSetting" + str(setting + 1) + "_Experim
```

```
        loss_hist_train, loss_hist_valid, accuracy_hist_train, accuracy_hist_valid = t
                                                                        dataLoader_inc
                                                                        dataLoader_inc
                                                                                   num
        plotModelPerformance(modelNum = 2, dataSettingNum = setting + 1,
                             loss_hist_train = loss_hist_train, loss_hist_valid = loss_his
                             accuracy_hist_train = accuracy_hist_train, accuracy_hist_vali
```

```
Experiment: 0 Setting: 0
N: 200
```

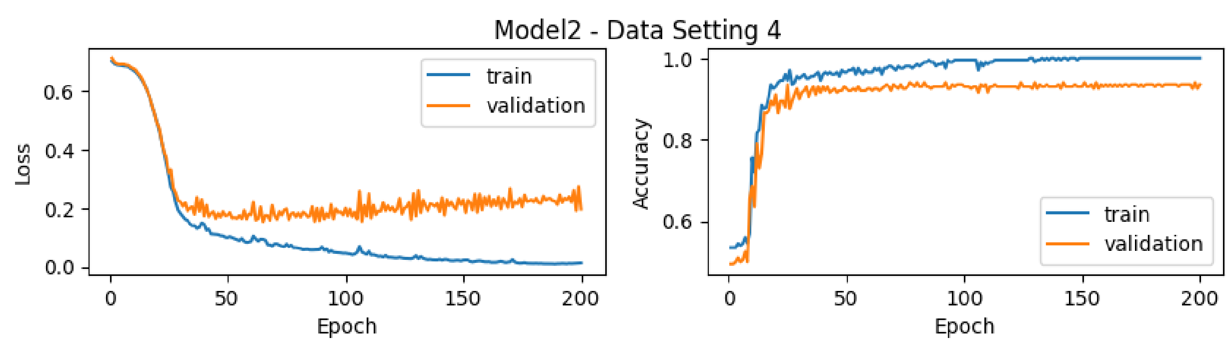



```
Experiment: 0 Setting: 1
N: 500
```

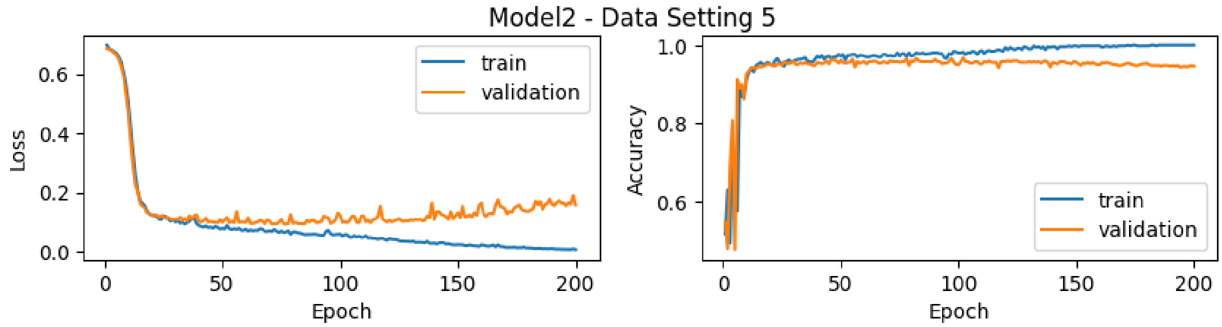



```
Experiment: 0 Setting: 2
N: 1000
```

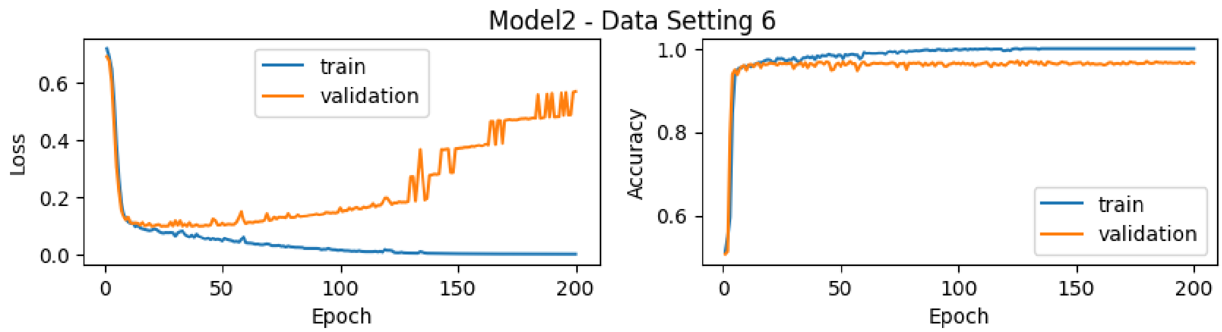

```
Experiment: 0 Setting: 3
N: 200
```



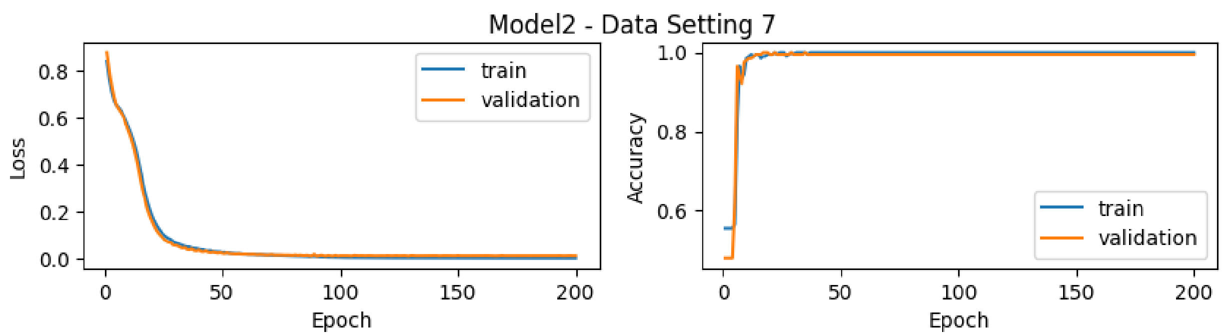
```
Experiment: 0 Setting: 4
N: 500
```



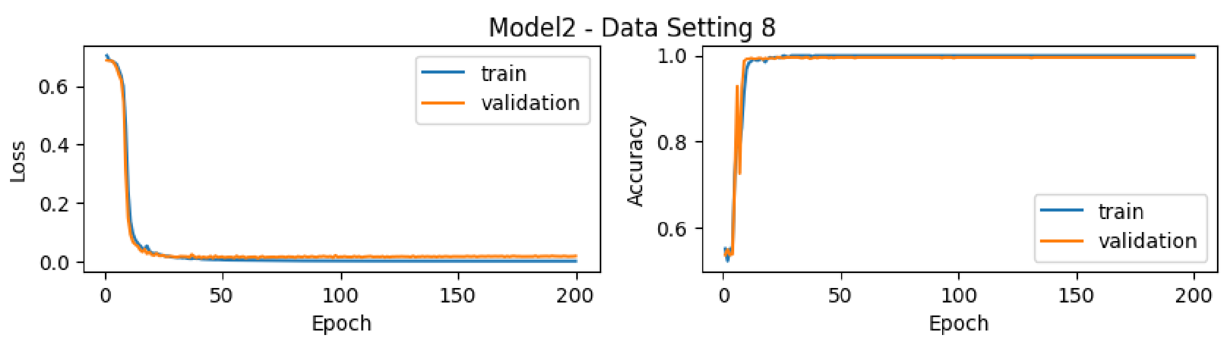

```
Experiment: 0 Setting: 5
N: 1000
```



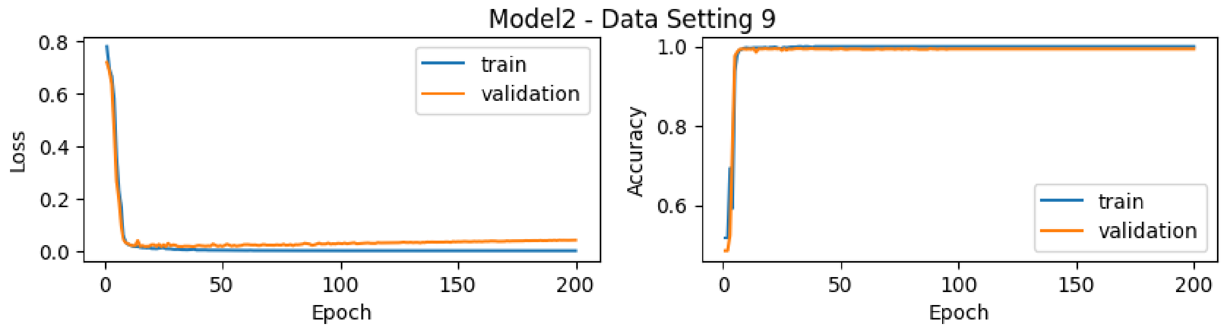

```
Experiment: 0 Setting: 6
N: 200
```



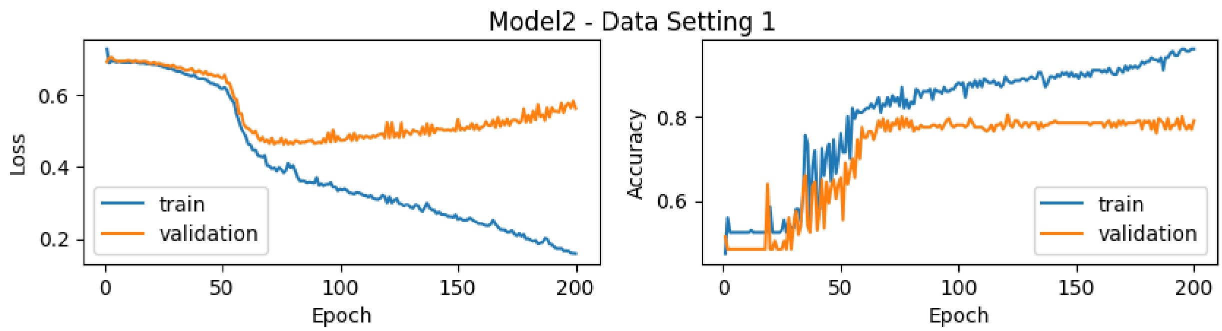

```
Experiment: 0 Setting: 7
N: 500
```



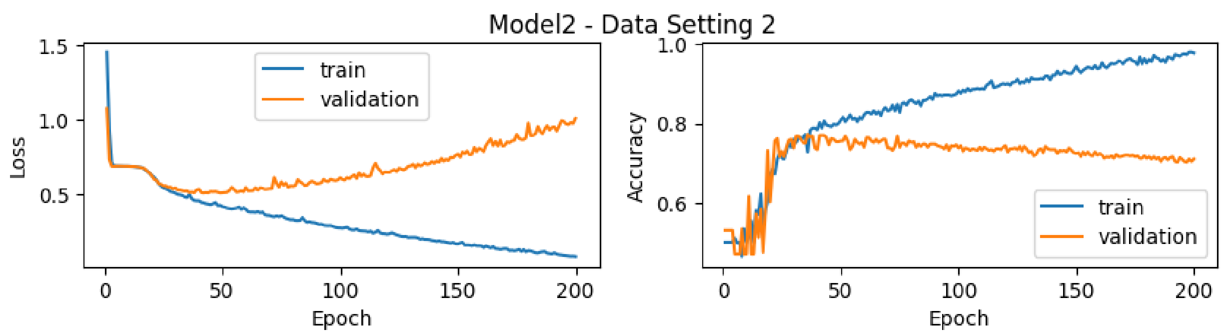

```
Experiment: 0 Setting: 8
N: 1000
```



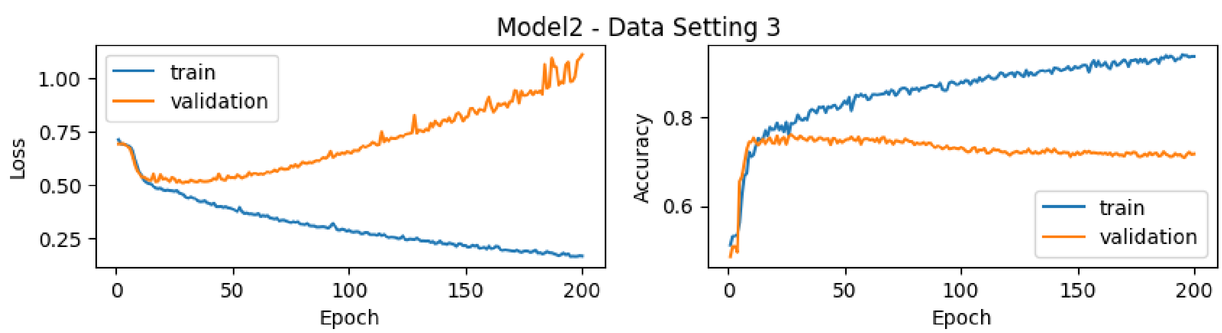

```
Experiment: 1 Setting: 0
N: 200
```



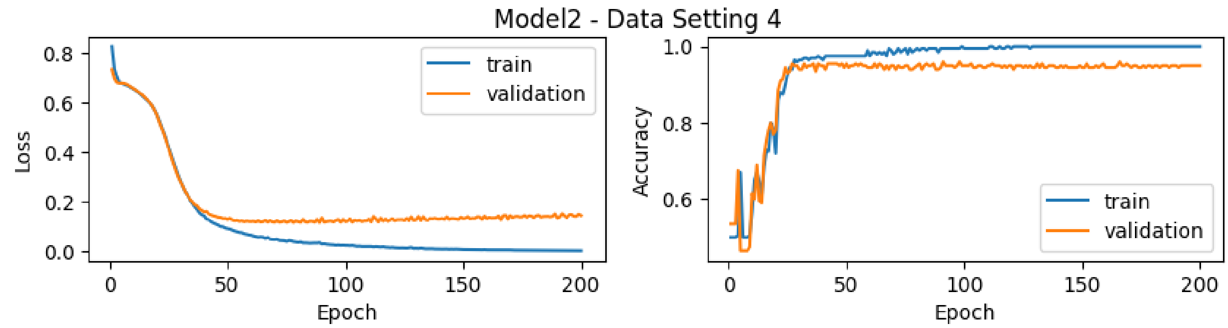

```
Experiment: 1 Setting: 1
N: 500
```



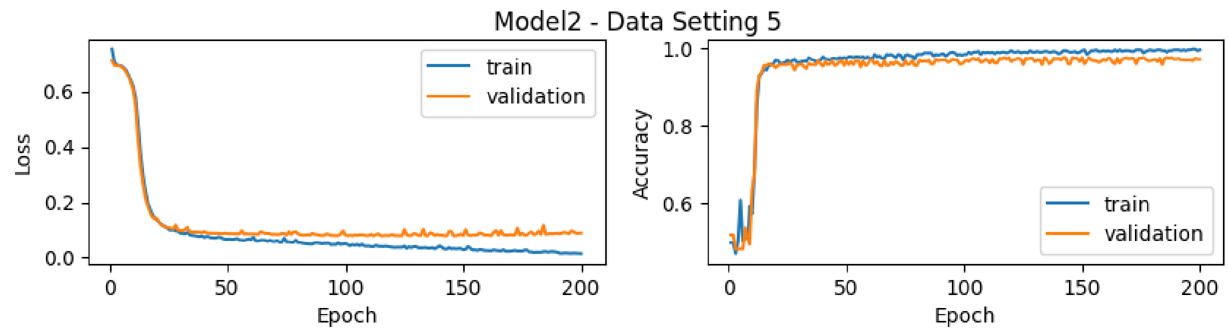

```
Experiment: 1 Setting: 2
N: 1000
```



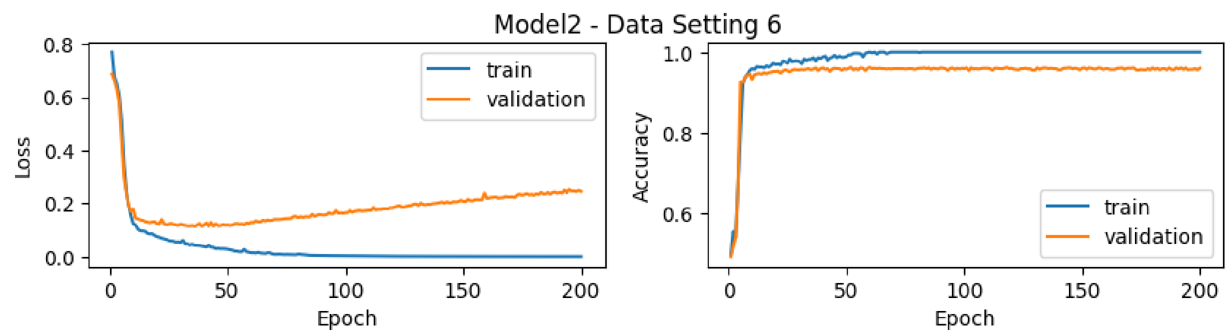

```
Experiment: 1 Setting: 3
N: 200
```



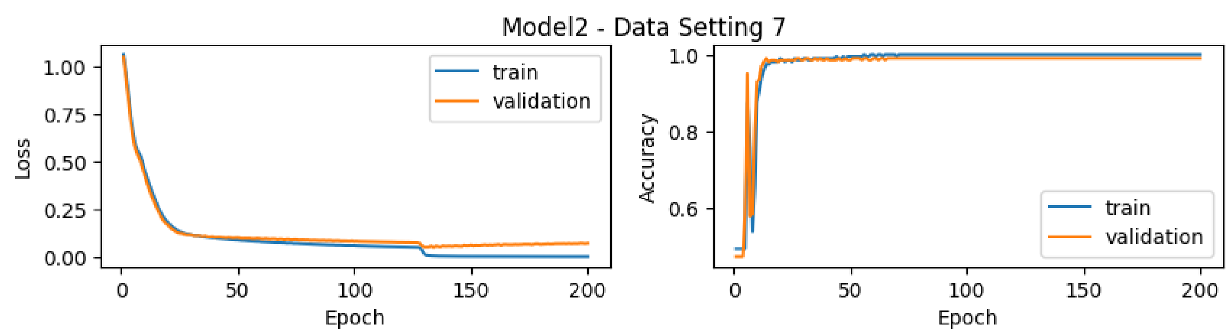

```
Experiment: 1 Setting: 4
N: 500
```



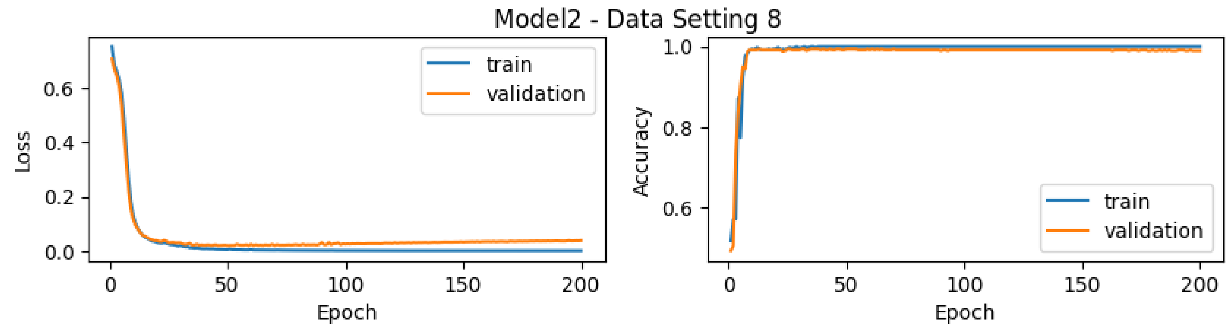

```
Experiment: 1 Setting: 5
N: 1000
```



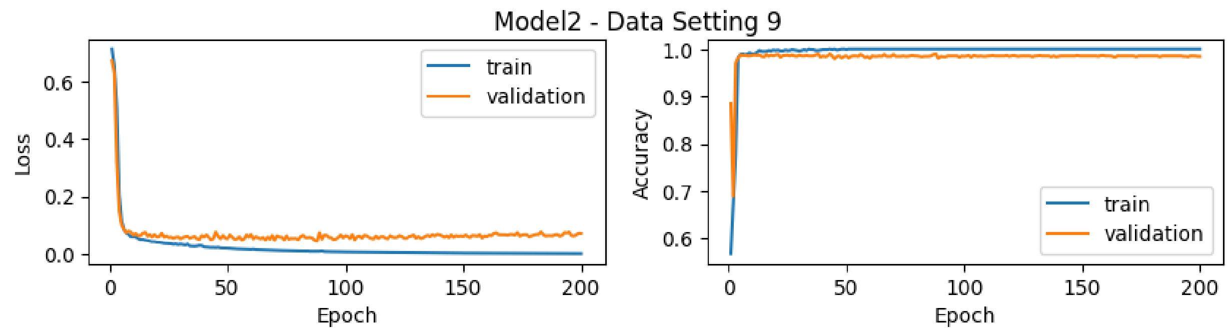

```
Experiment: 1 Setting: 6
N: 200
```



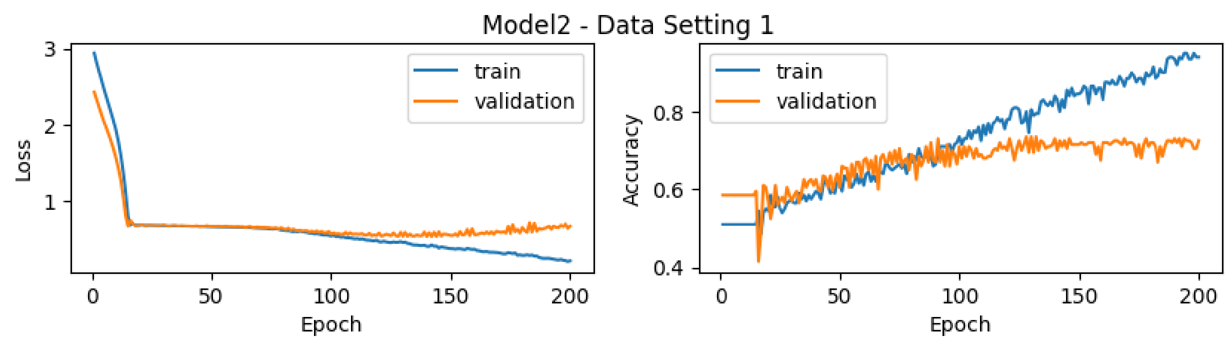

```
Experiment: 1 Setting: 7
N: 500
```



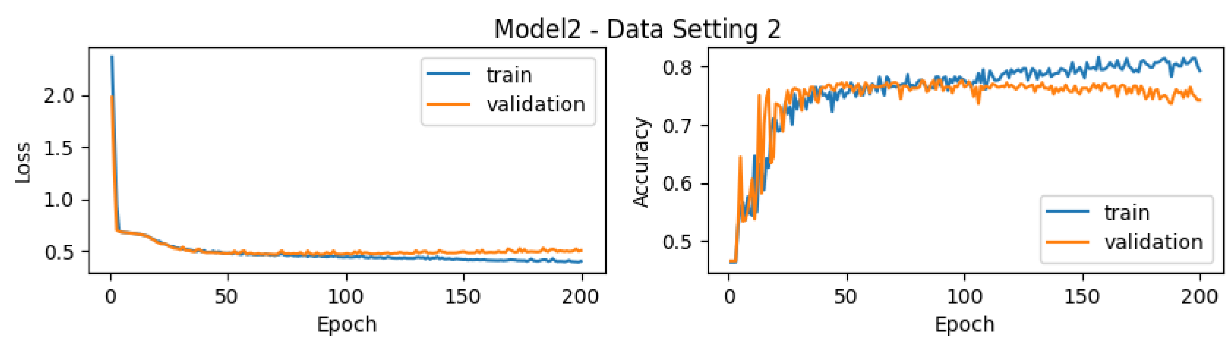

```
Experiment: 1 Setting: 8
N: 1000
```



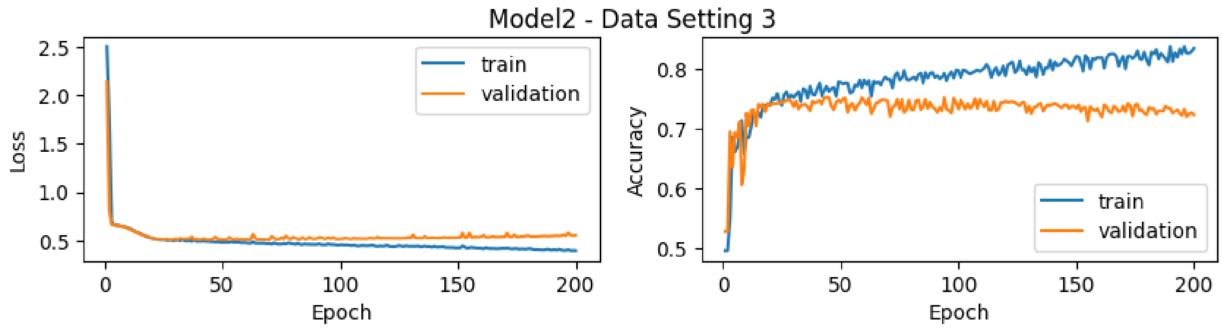

```
Experiment: 2 Setting: 0
N: 200
```



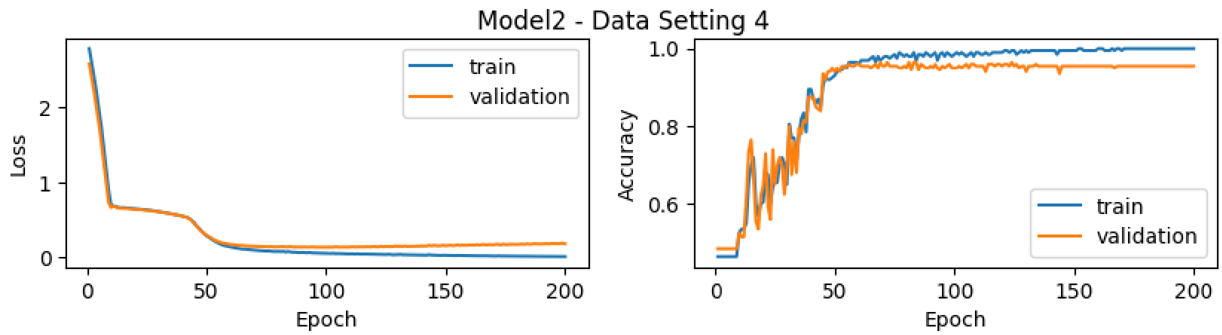

```
Experiment: 2 Setting: 1
N: 500
```



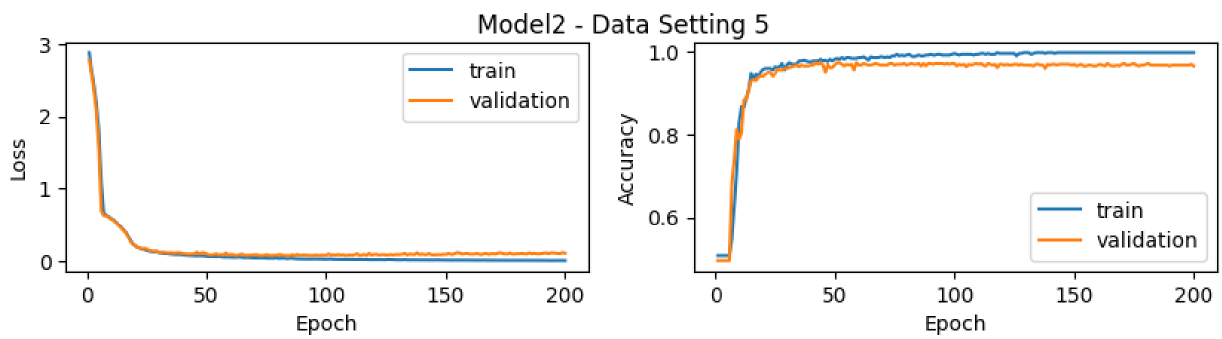

```
Experiment: 2 Setting: 2
N: 1000
```



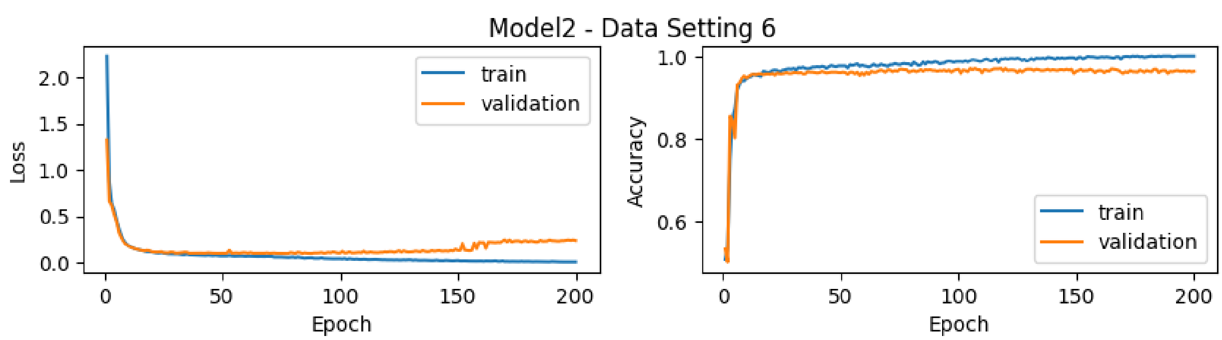

```
Experiment: 2 Setting: 3
N: 200
```



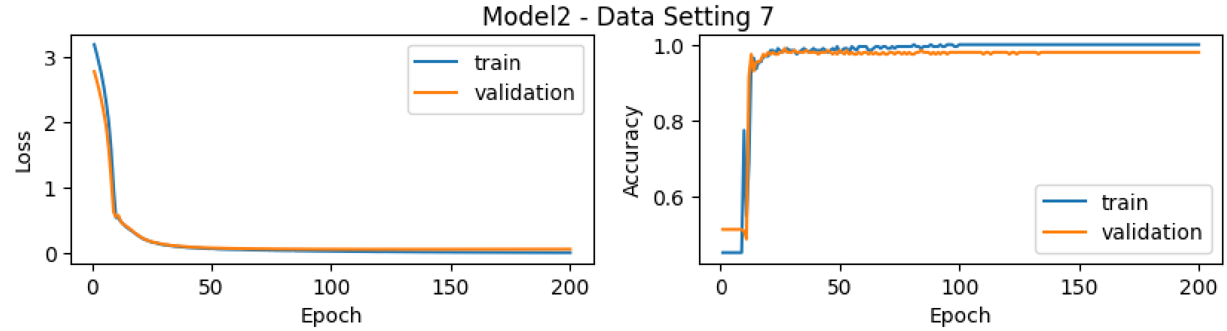

```
Experiment: 2 Setting: 4
N: 500
```



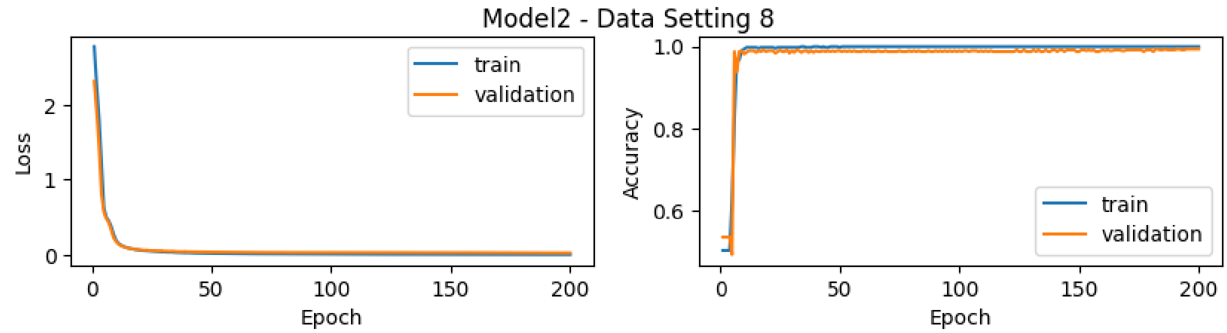

```
Experiment: 2 Setting: 5
N: 1000
```



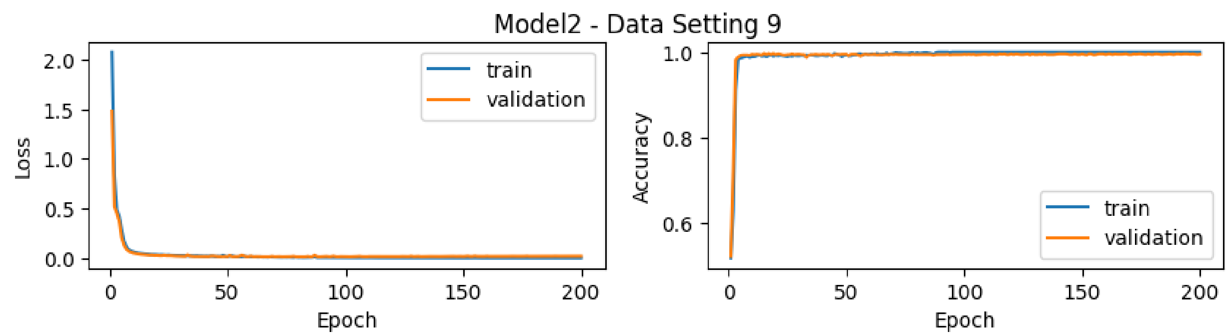

```
Experiment: 2 Setting: 6
N: 200
```



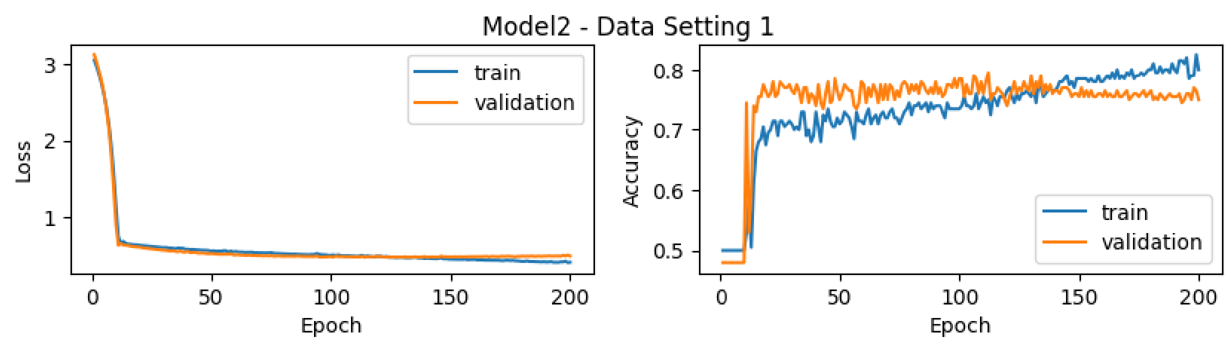

```
Experiment: 2 Setting: 7
N: 500
```



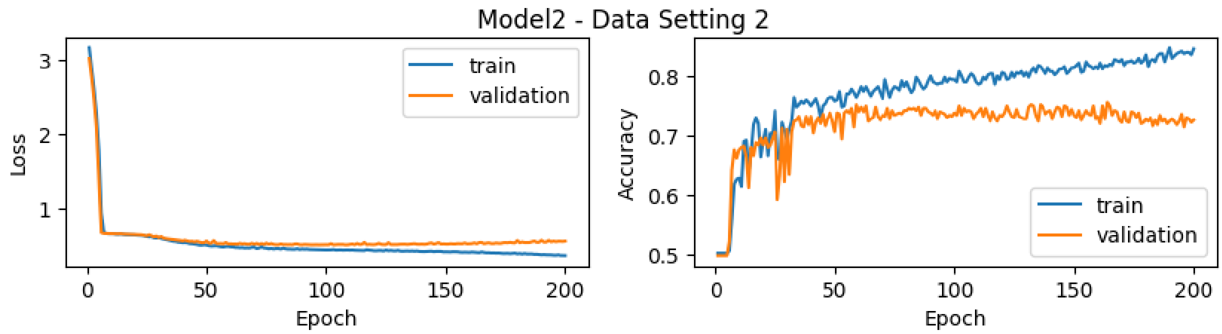

```
Experiment: 2 Setting: 8
N: 1000
```



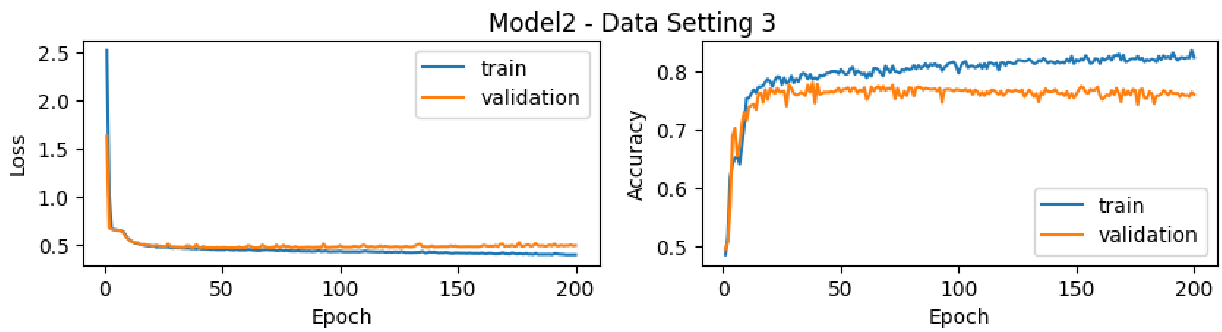

```
Experiment: 3 Setting: 0
N: 200
```



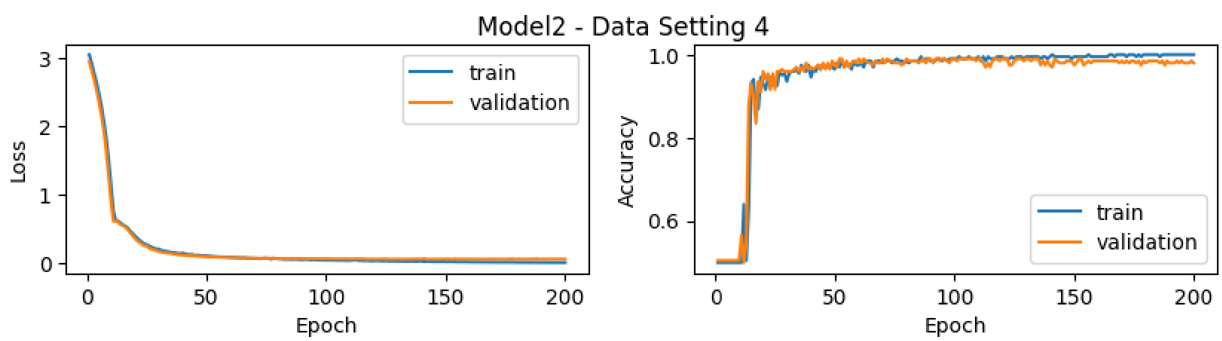

```
Experiment: 3 Setting: 1
N: 500
```



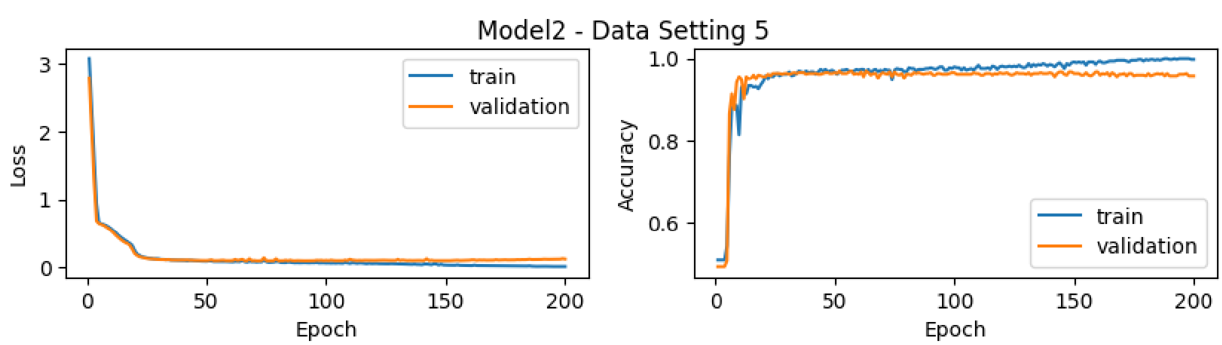

```
Experiment: 3 Setting: 2
N: 1000
```



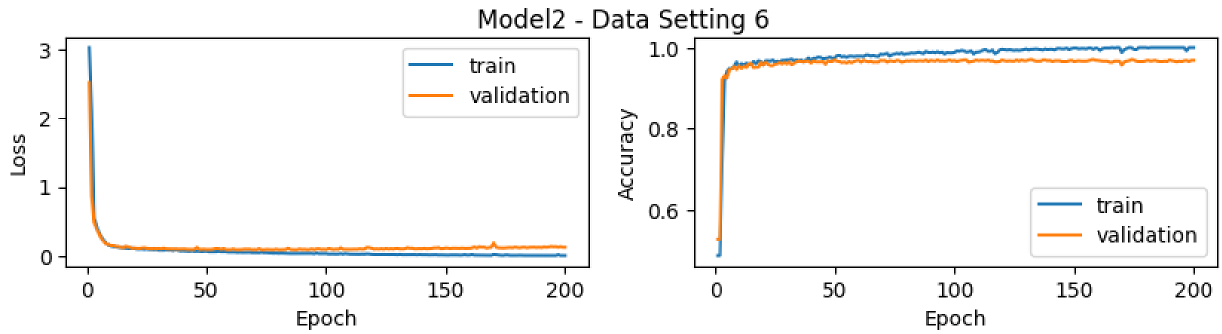

```
Experiment: 3 Setting: 3
N: 200
```



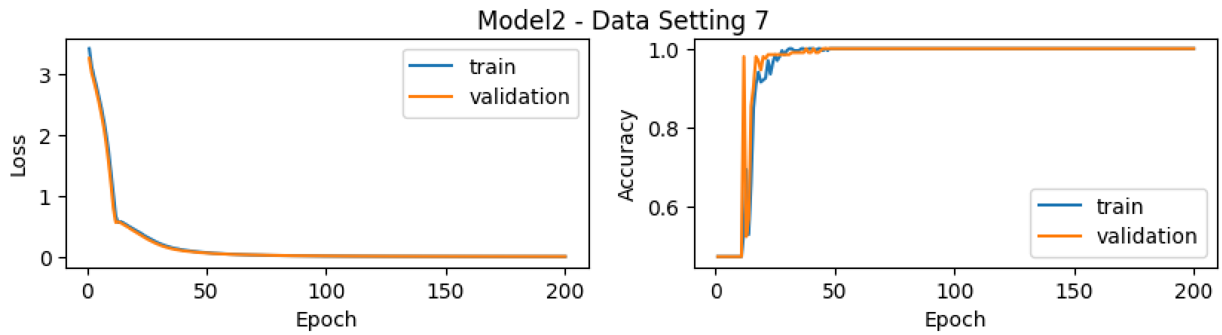

```
Experiment: 3 Setting: 4
N: 500
```



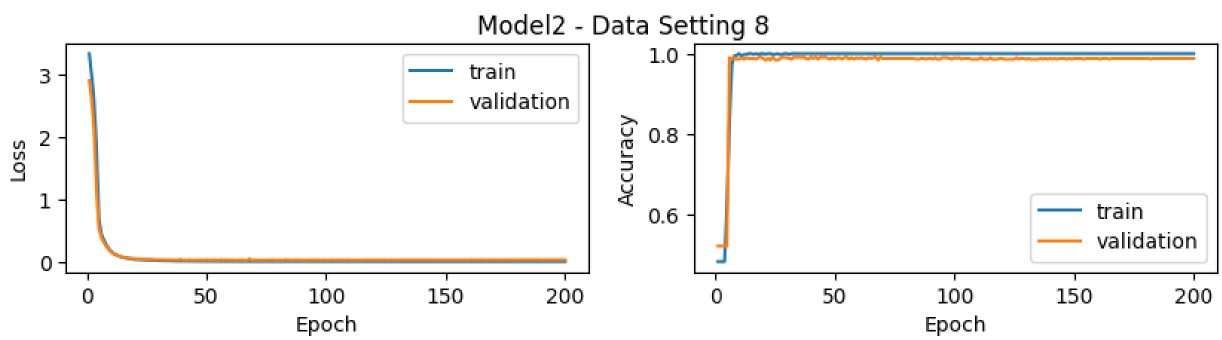

```
Experiment: 3 Setting: 5
N: 1000
```



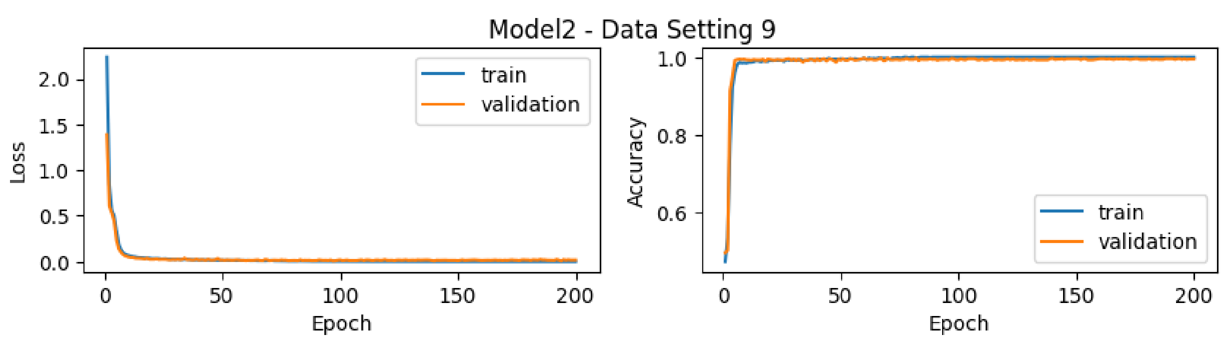

```
Experiment: 3 Setting: 6
N: 200
```



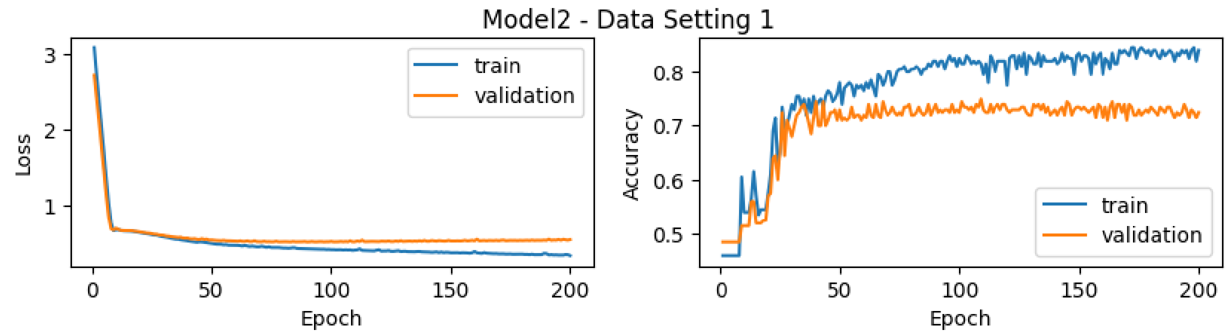

```
Experiment: 3 Setting: 7
N: 500
```



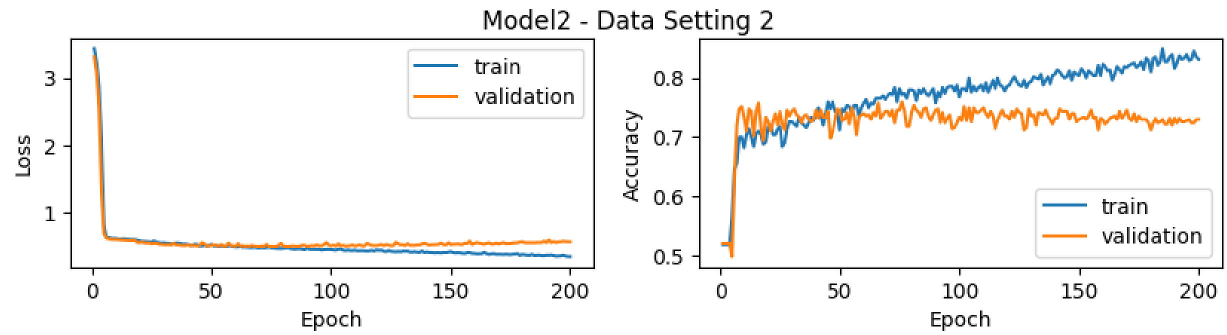

```
Experiment: 3 Setting: 8
N: 1000
```



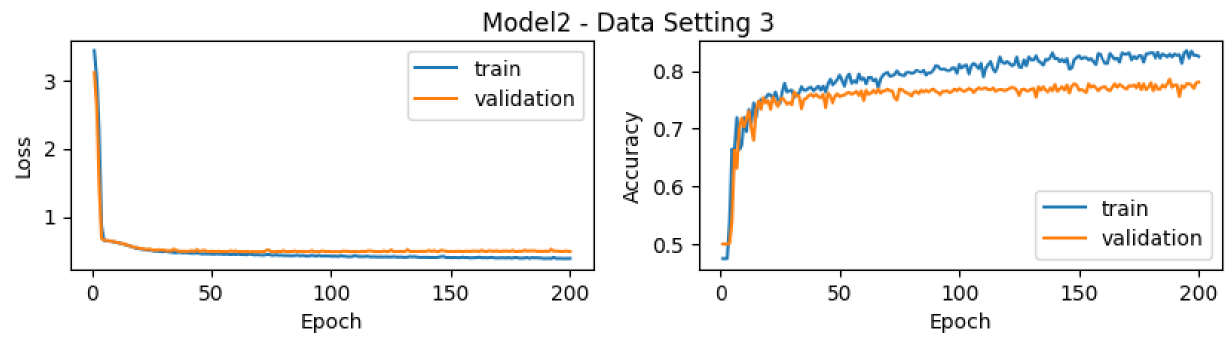

```
Experiment: 4 Setting: 0
N: 200
```



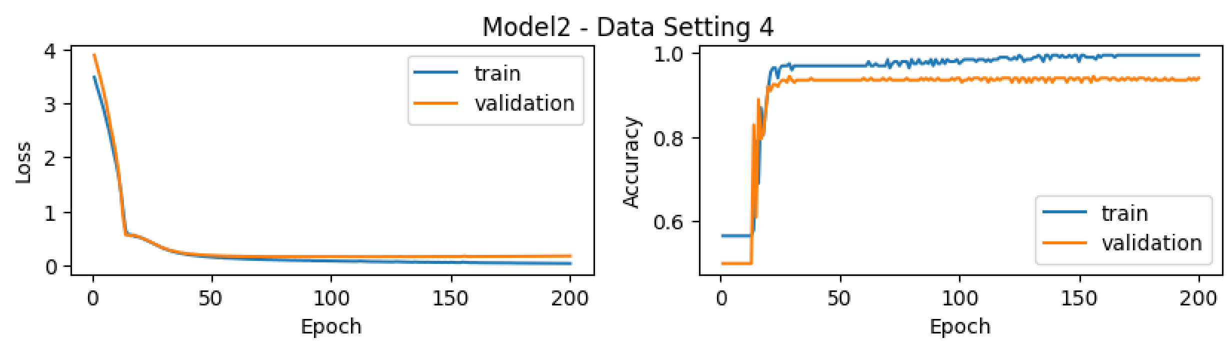

```
Experiment: 4 Setting: 1
N: 500
```



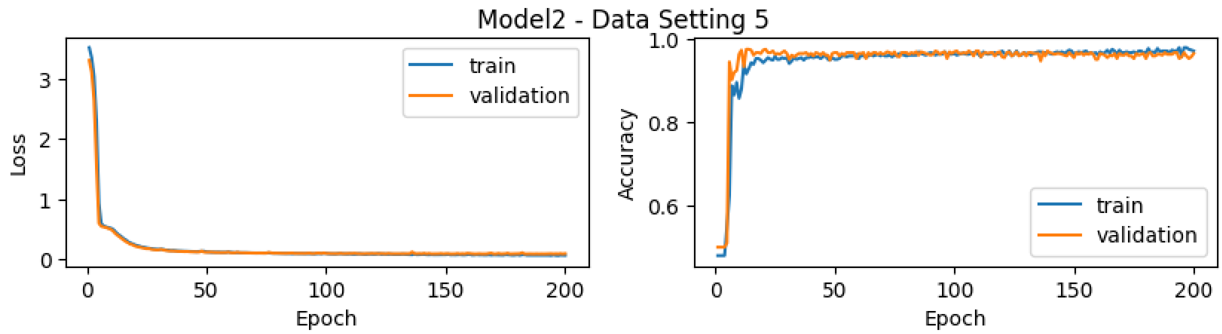

```
Experiment: 4 Setting: 2
N: 1000
```



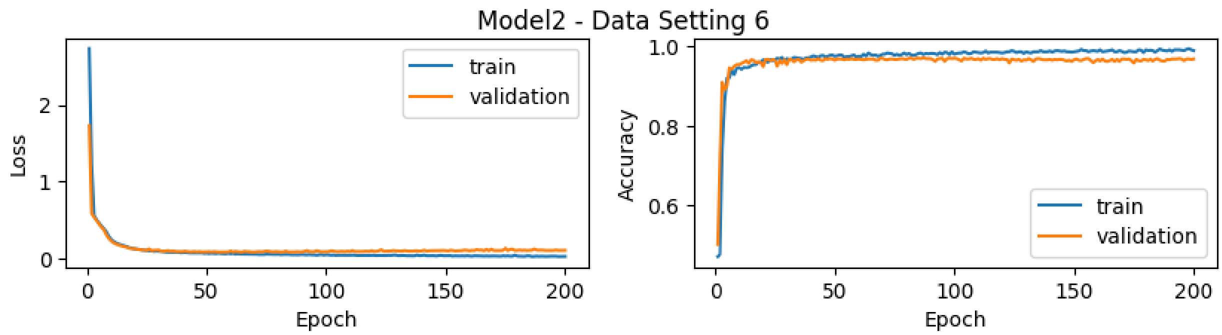

```
Experiment: 4 Setting: 3
N: 200
```



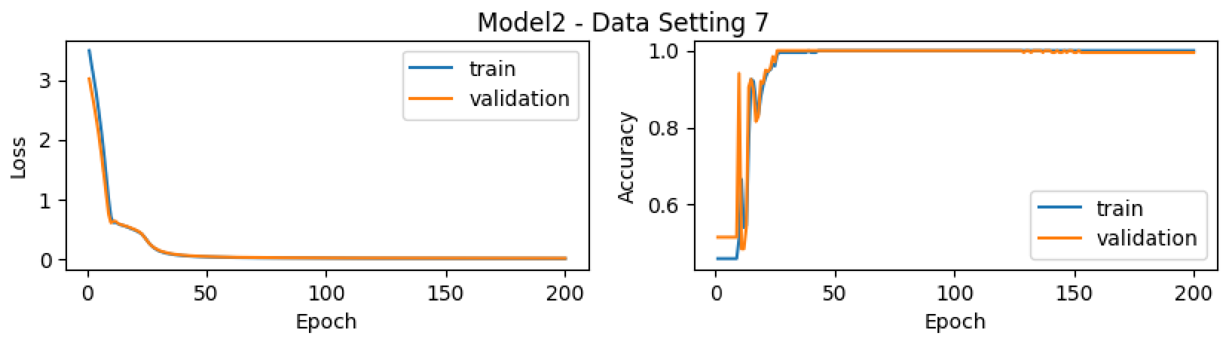

```
Experiment: 4 Setting: 4
N: 500
```



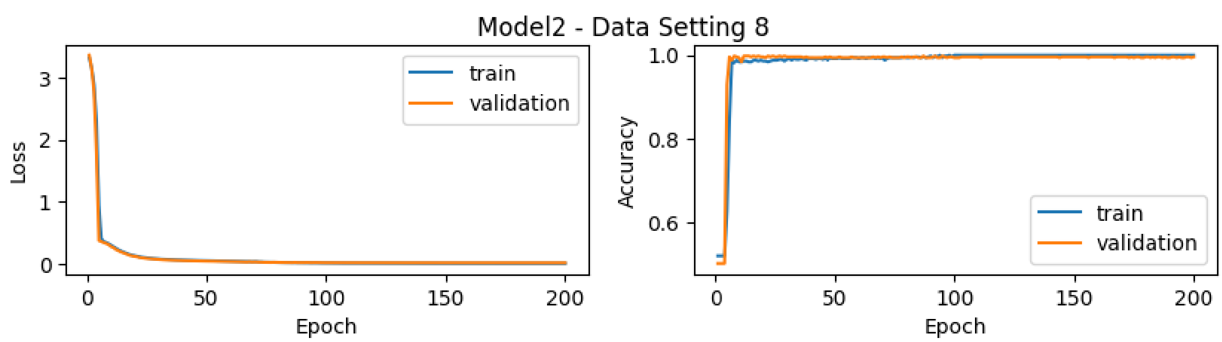

```
Experiment: 4 Setting: 5
N: 1000
```




```
Experiment: 4 Setting: 6
N: 200
```

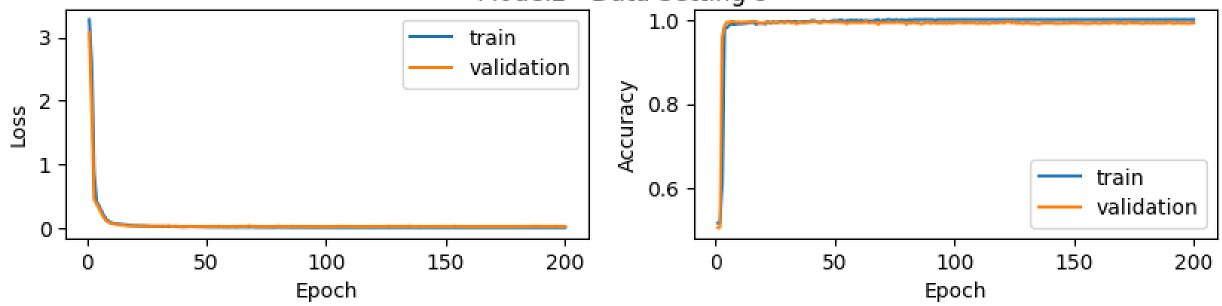



```
Experiment: 4 Setting: 7
N: 500
```



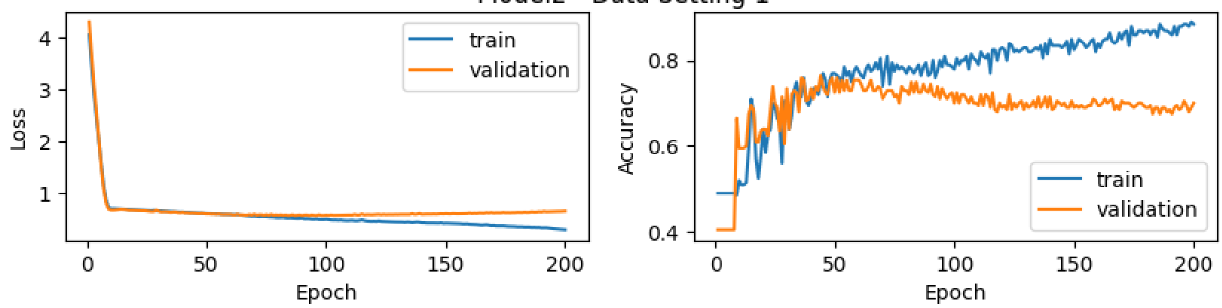

```
Experiment: 4 Setting: 8
N: 1000
```

Model2 - Data Setting 9



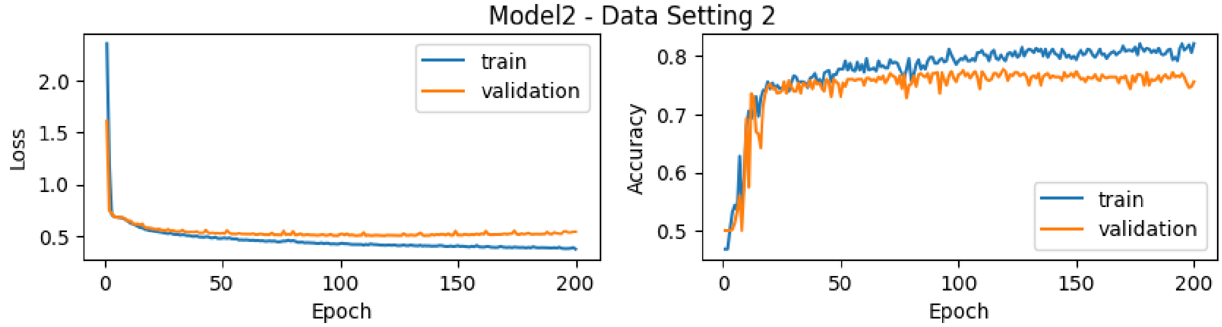

```
Experiment: 5 Setting: 0
N: 200
```

Model2 - Data Setting 1



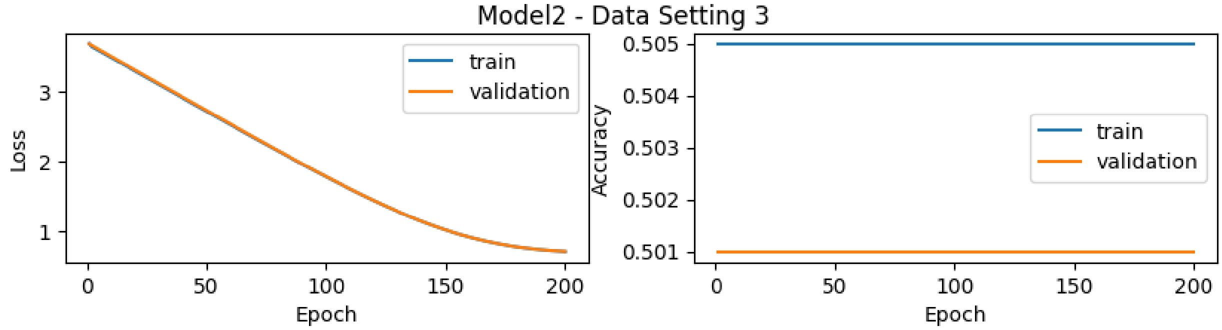

```
Experiment: 5 Setting: 1
N: 500
```

Model2 - Data Setting 2

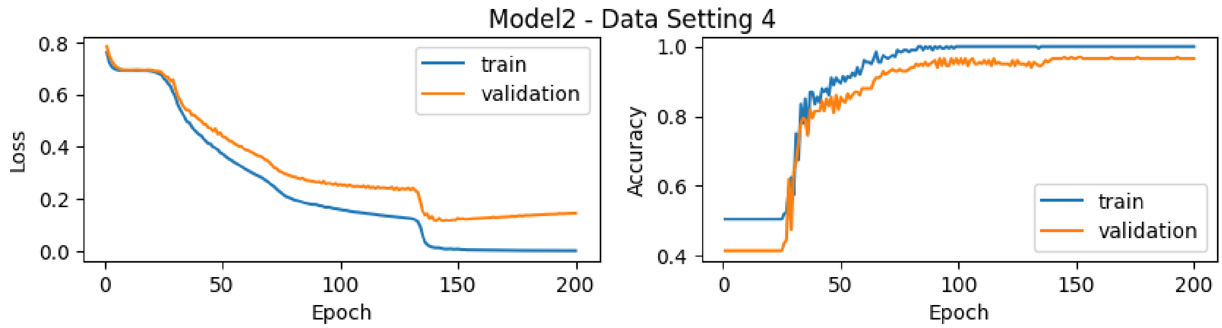



```
Experiment: 5 Setting: 2
N: 1000
```

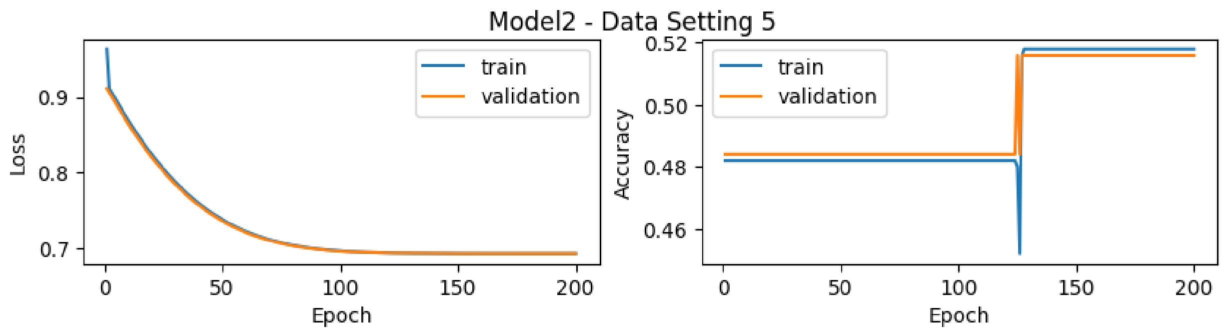
Model2 - Data Setting 3



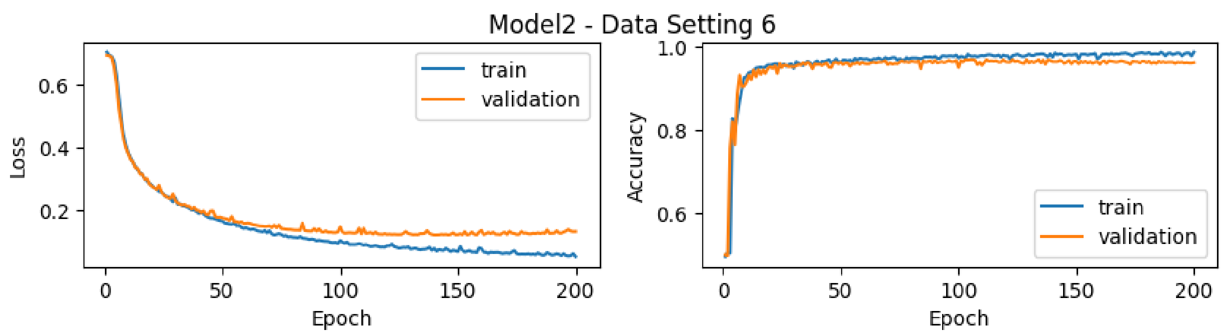

```
Experiment: 5 Setting: 3
N: 200
```



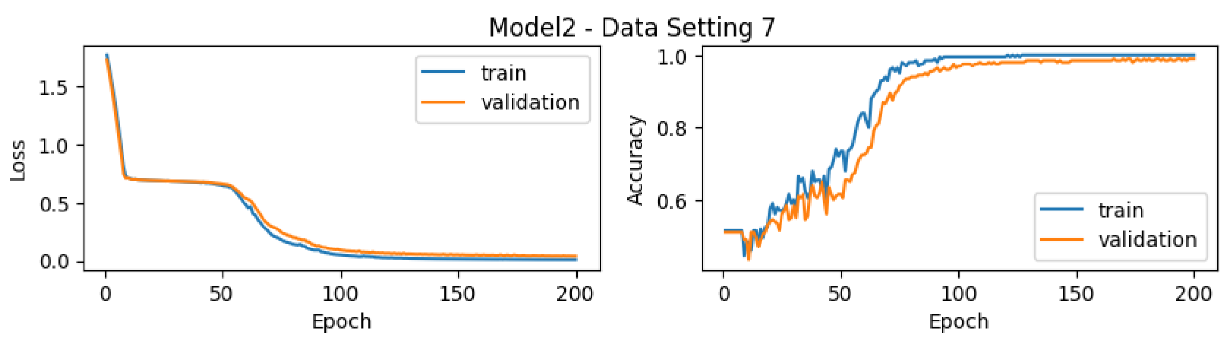

```
Experiment: 5 Setting: 4
N: 500
```



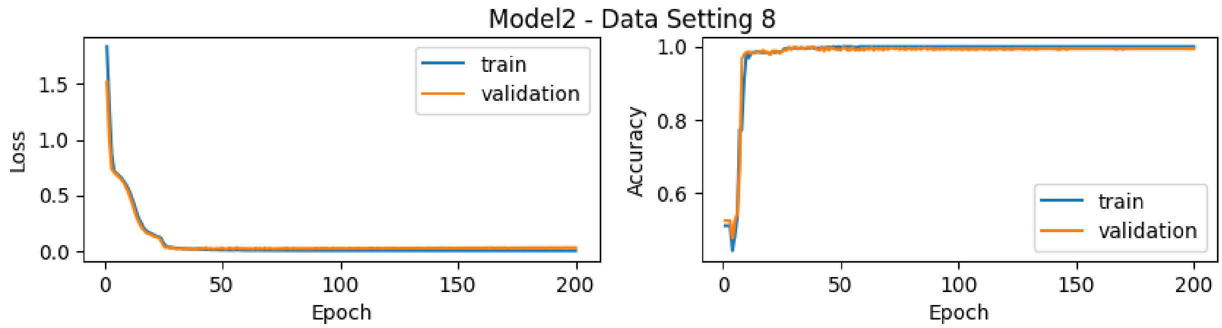

```
Experiment: 5 Setting: 5
N: 1000
```



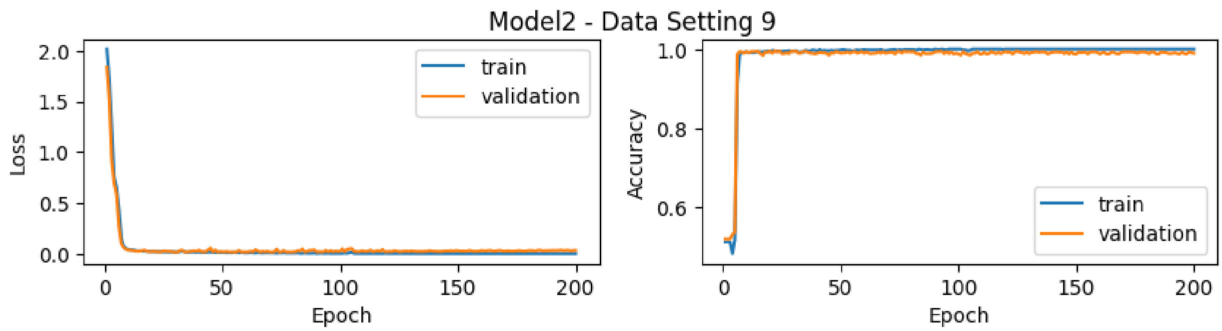

```
Experiment: 5 Setting: 6
N: 200
```



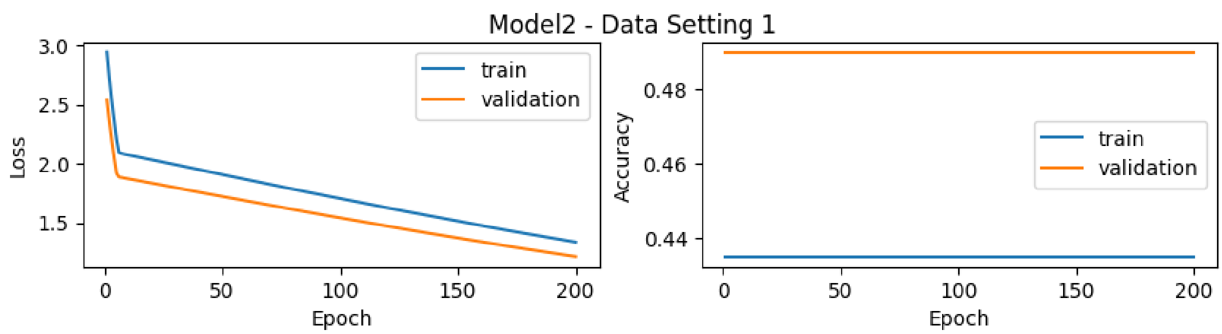

```
Experiment: 5 Setting: 7
N: 500
```



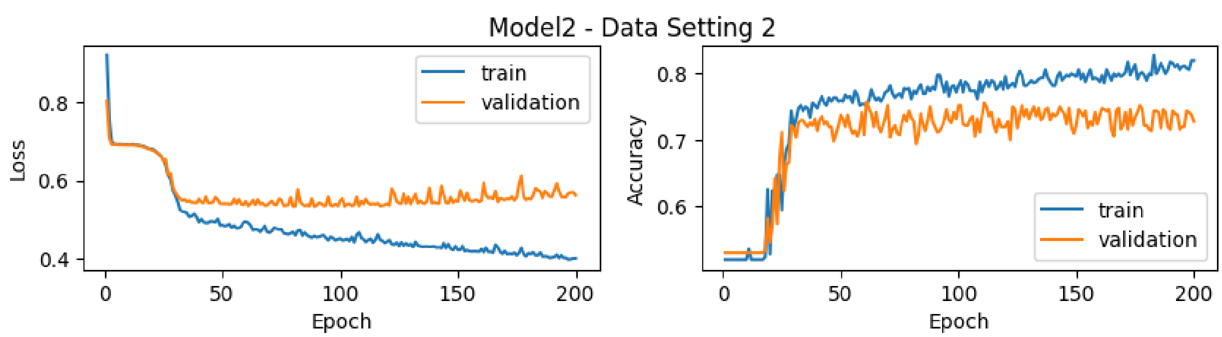

```
Experiment: 5 Setting: 8
N: 1000
```



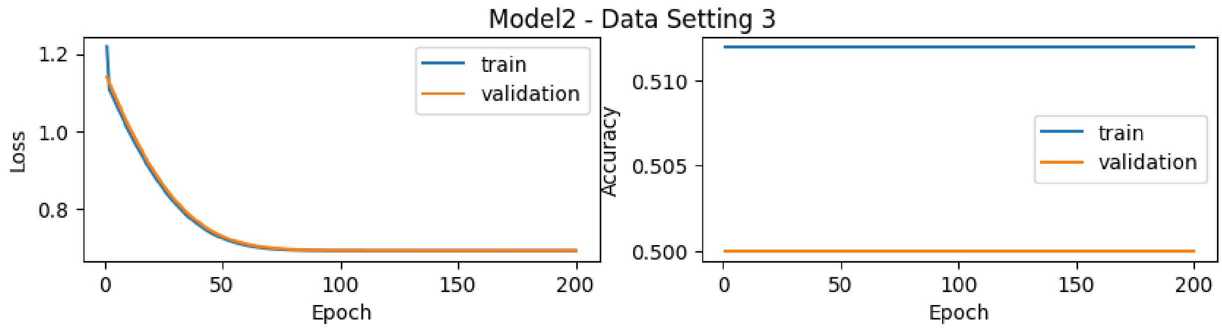

```
Experiment: 6 Setting: 0
N: 200
```



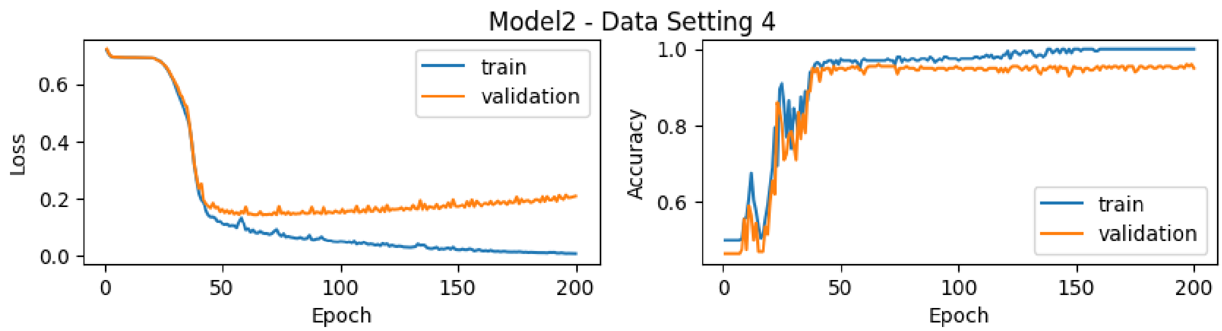

```
Experiment: 6 Setting: 1
N: 500
```



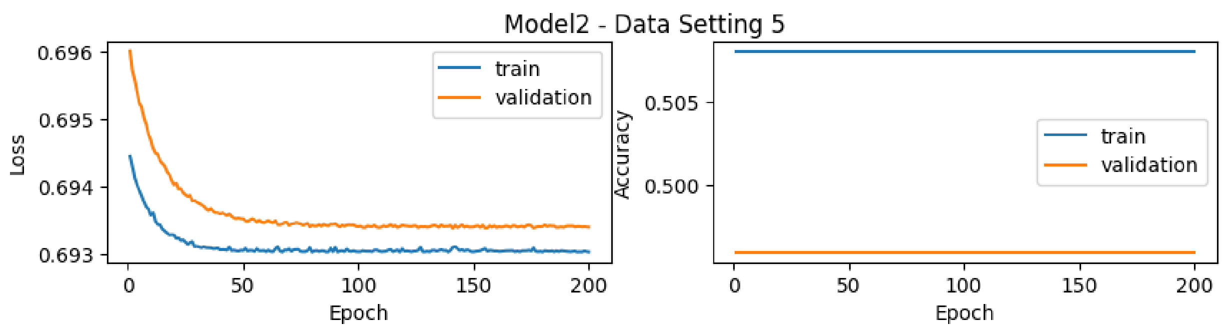

```
Experiment: 6 Setting: 2
N: 1000
```



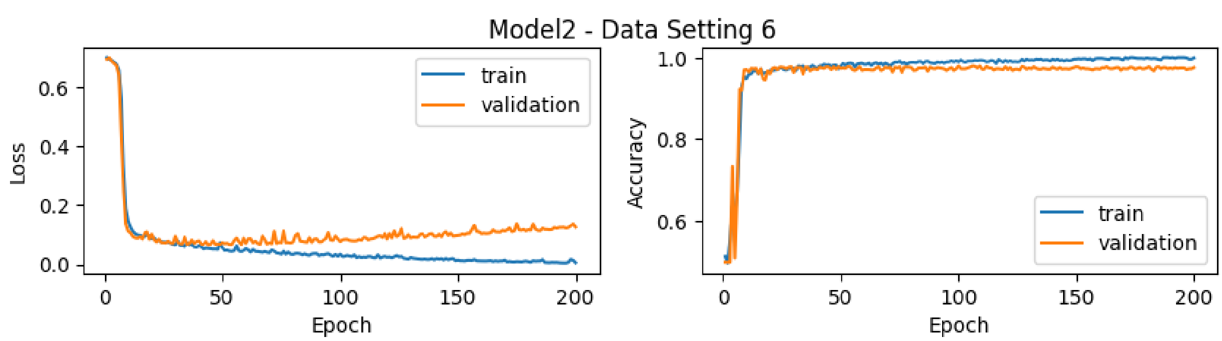

```
Experiment: 6 Setting: 3
N: 200
```



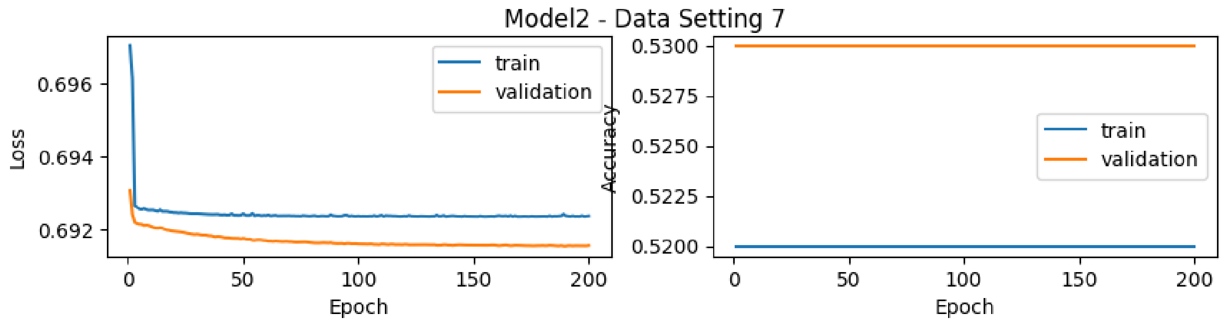

```
Experiment: 6 Setting: 4
N: 500
```



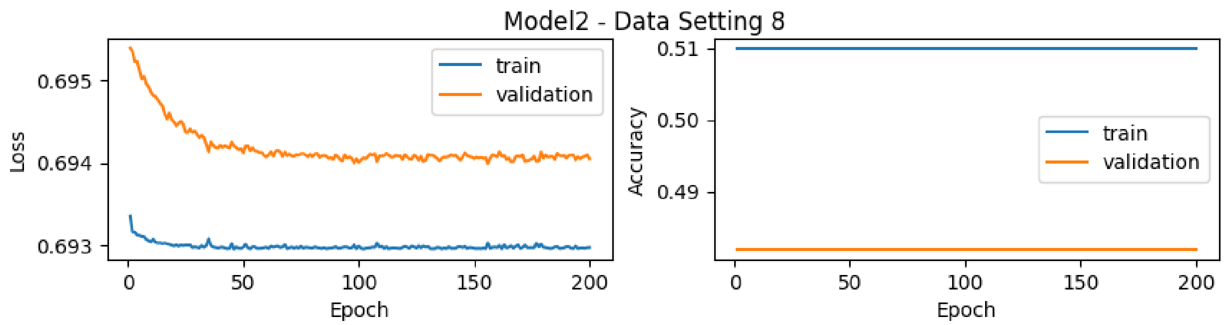

```
Experiment: 6 Setting: 5
N: 1000
```



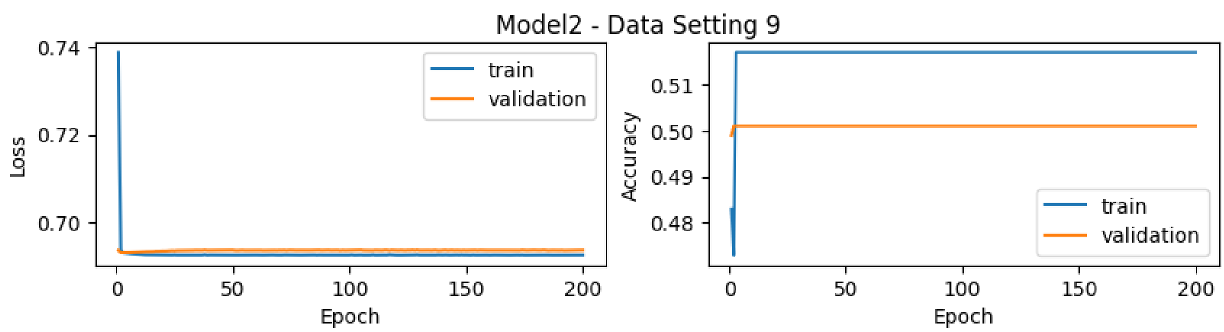

```
Experiment: 6 Setting: 6
N: 200
```



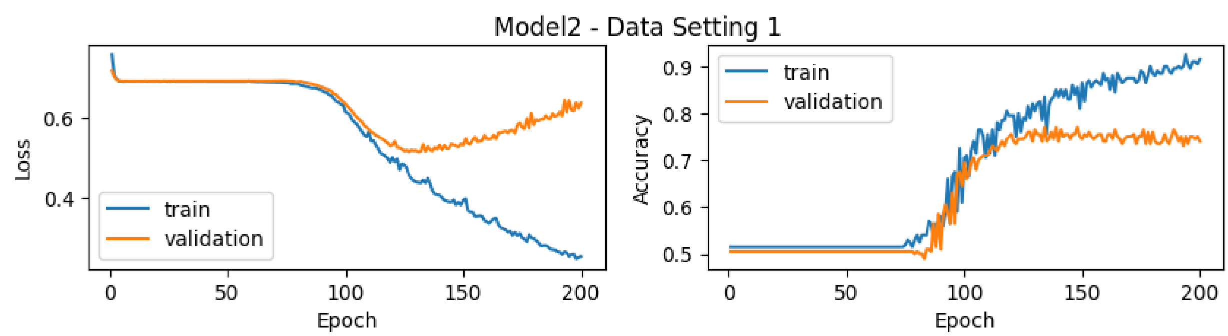

```
Experiment: 6 Setting: 7
N: 500
```



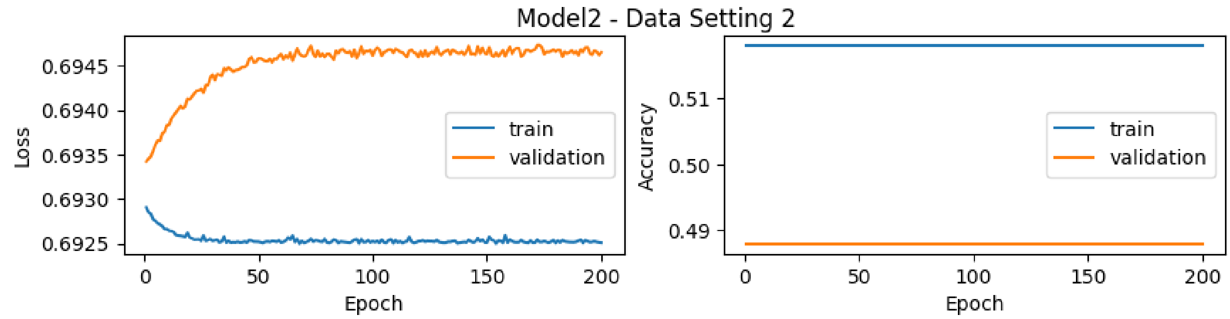

```
Experiment: 6 Setting: 8
N: 1000
```



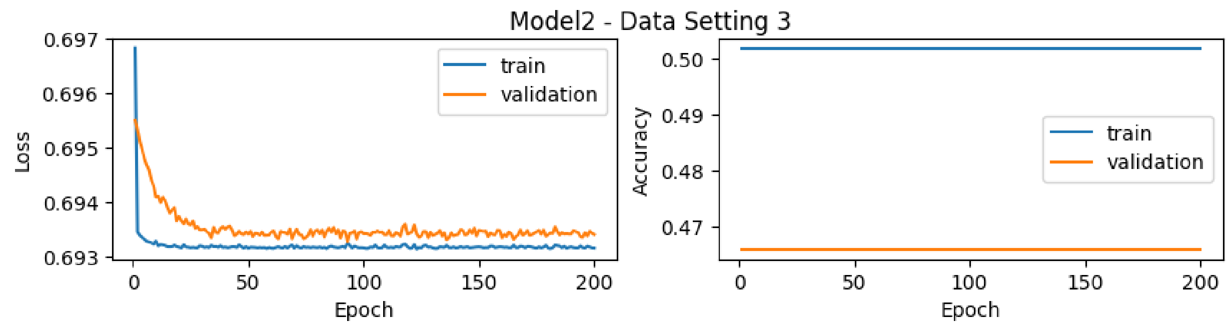

```
Experiment: 7 Setting: 0
N: 200
```



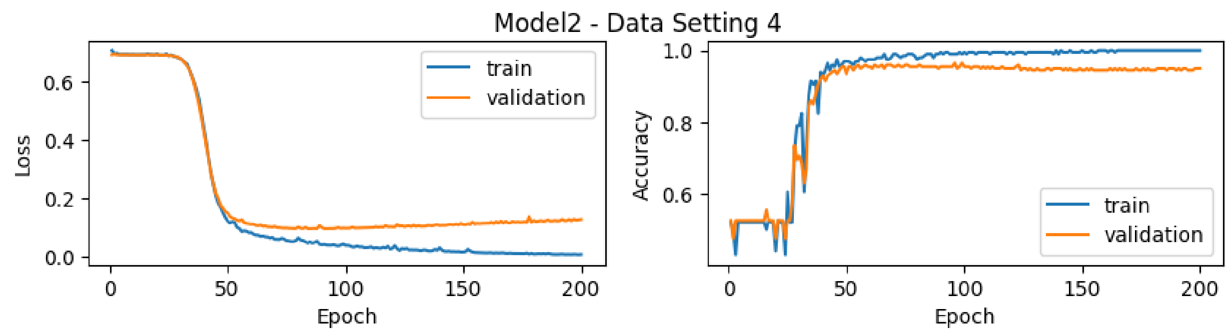

```
Experiment: 7 Setting: 1
N: 500
```



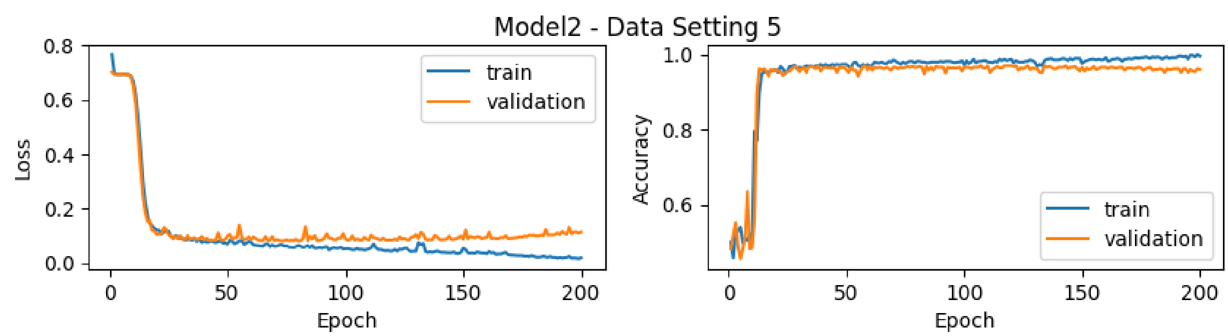

```
Experiment: 7 Setting: 2
N: 1000
```



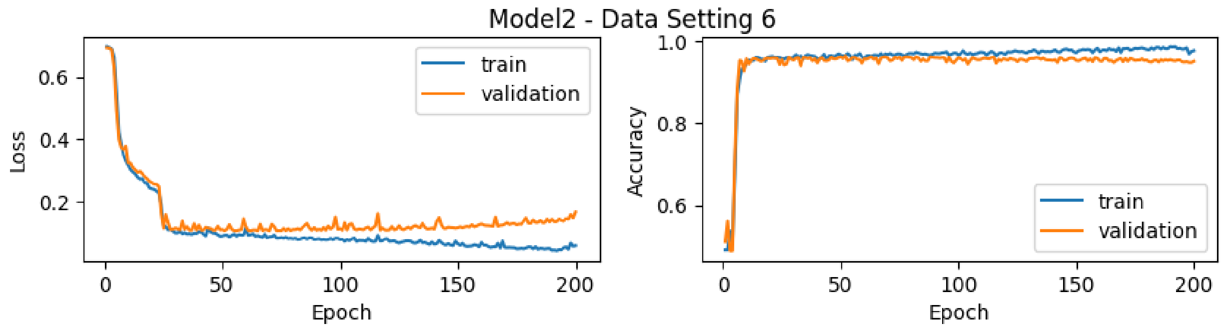

```
Experiment: 7 Setting: 3
N: 200
```



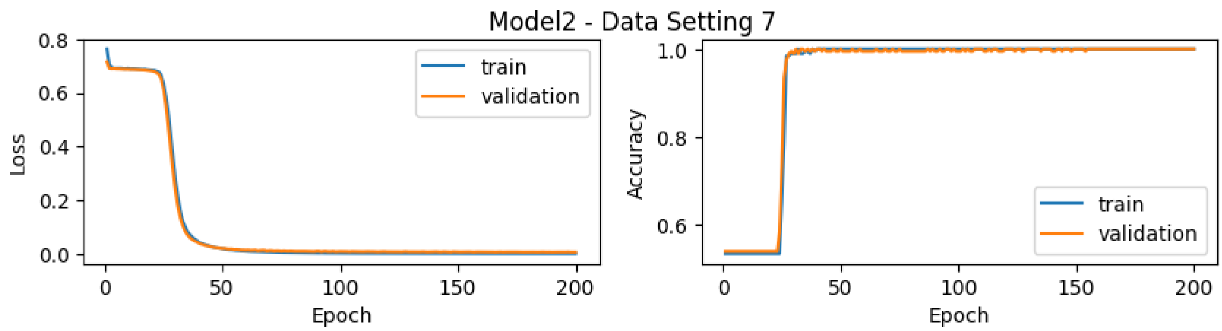

```
Experiment: 7 Setting: 4
N: 500
```



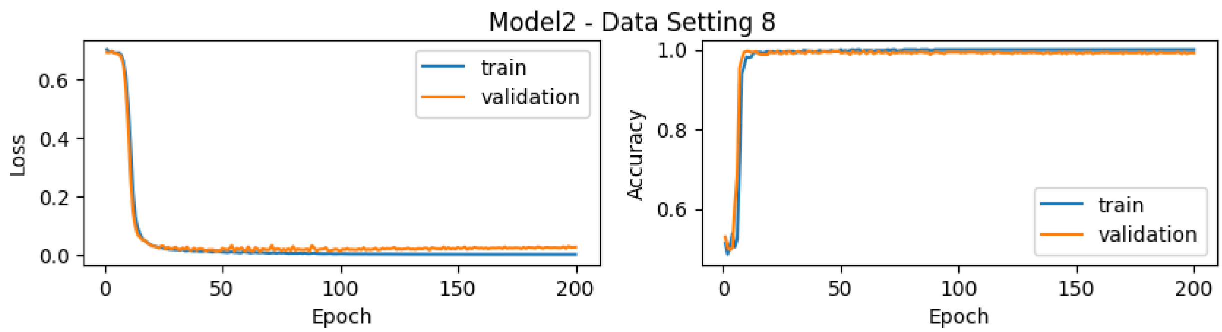

```
Experiment: 7 Setting: 5
N: 1000
```



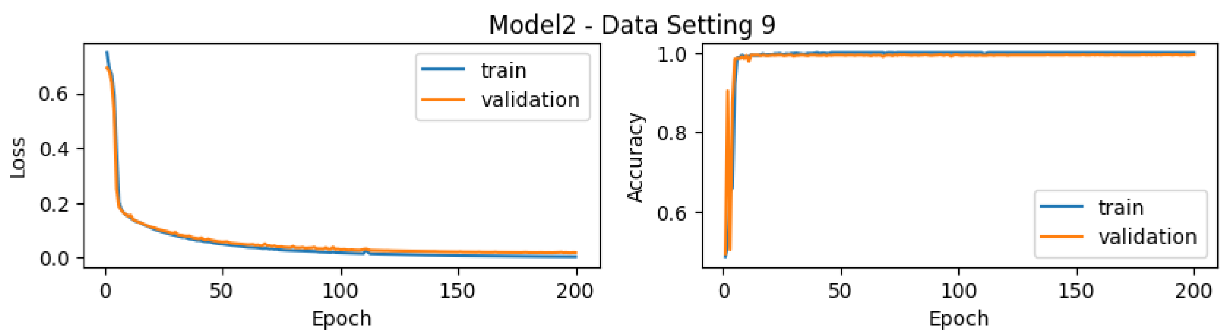

```
Experiment: 7 Setting: 6
N: 200
```



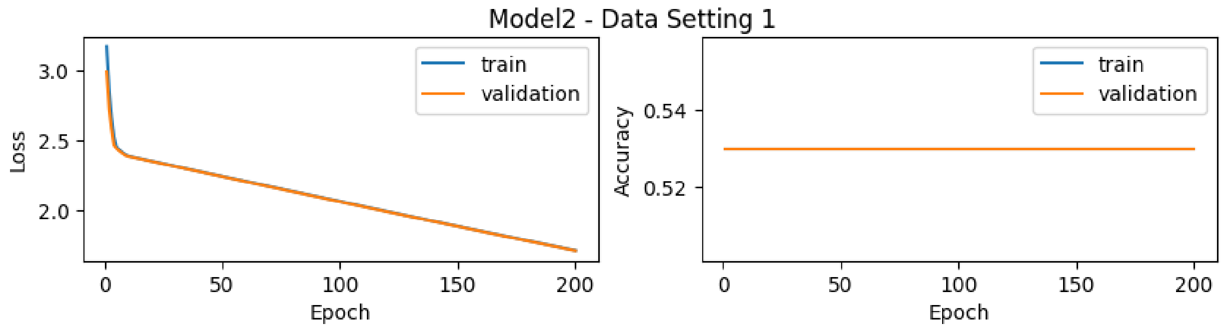

```
Experiment: 7 Setting: 7
N: 500
```



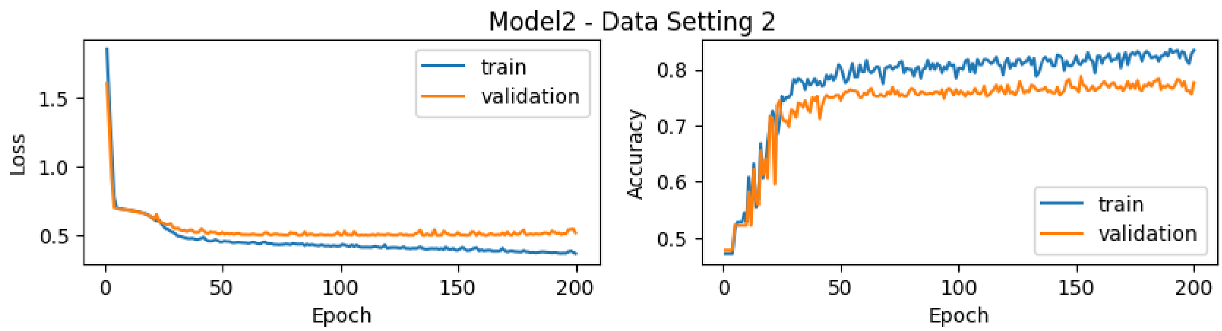

```
Experiment: 7 Setting: 8
N: 1000
```



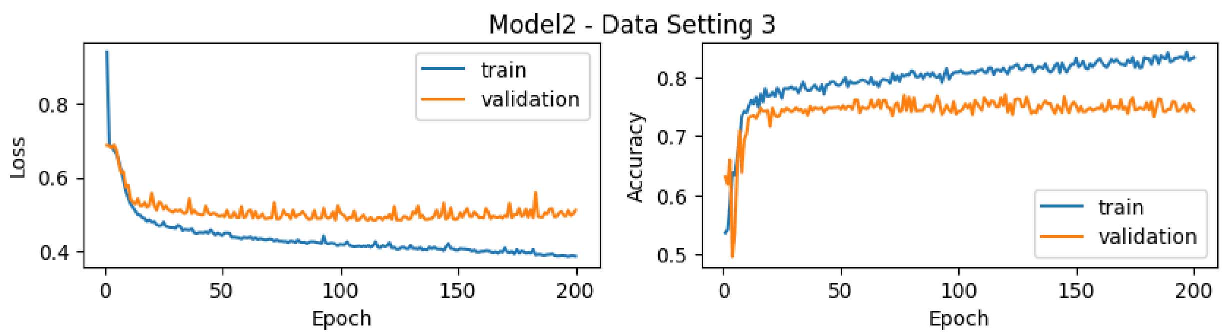

```
Experiment: 8 Setting: 0
N: 200
```



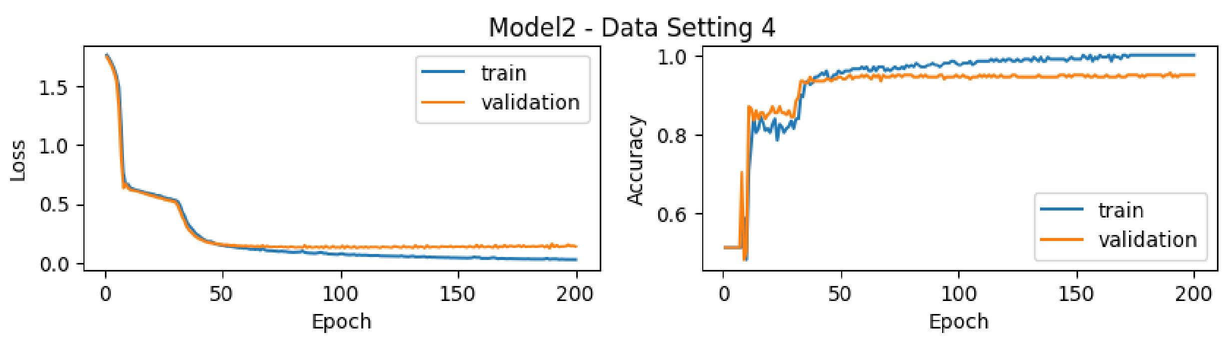

```
Experiment: 8 Setting: 1
N: 500
```



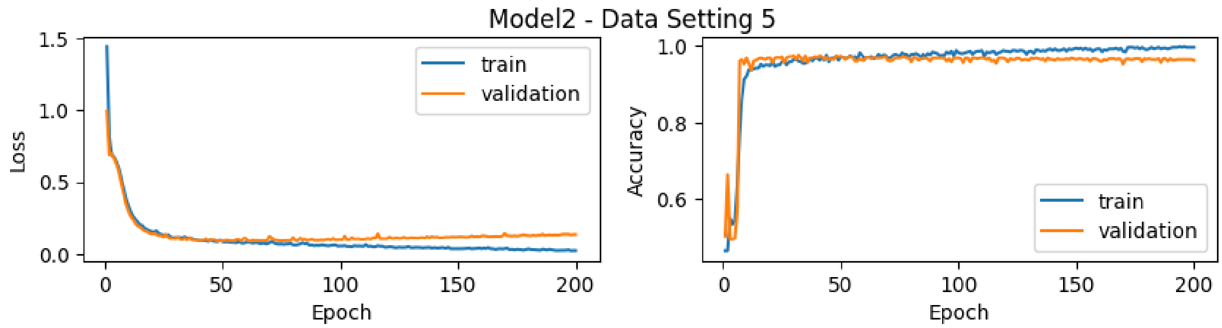

```
Experiment: 8 Setting: 2
N: 1000
```



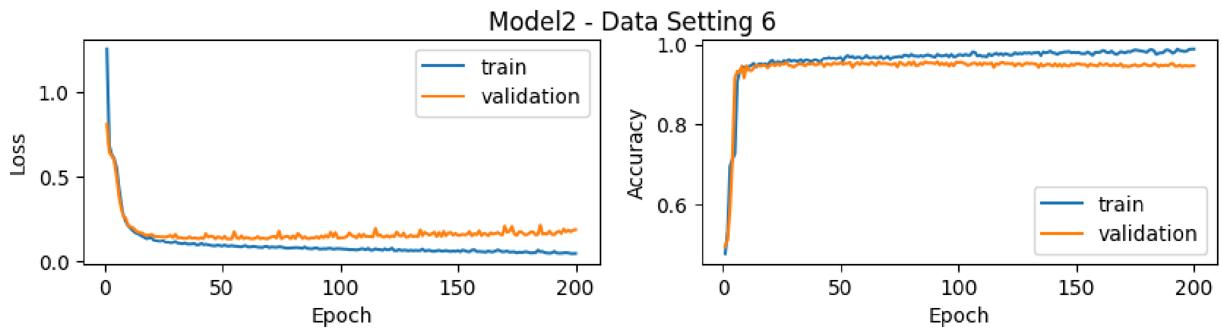

```
Experiment: 8 Setting: 3
N: 200
```



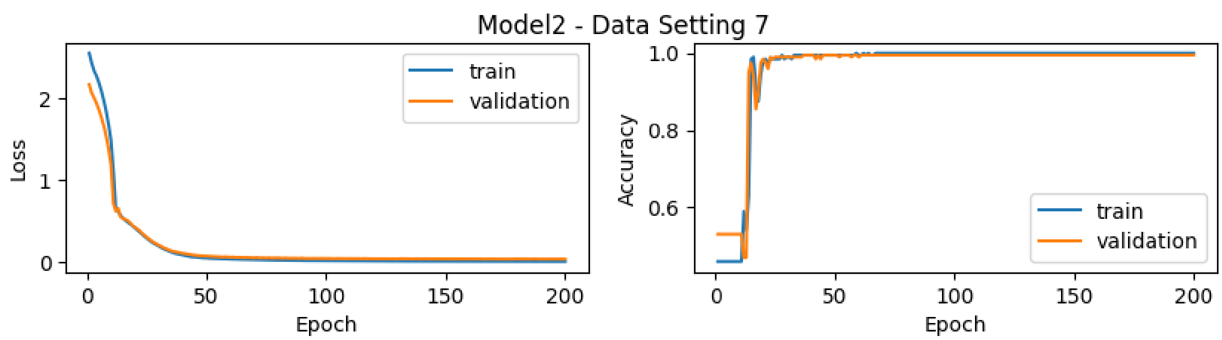

```
Experiment: 8 Setting: 4
N: 500
```



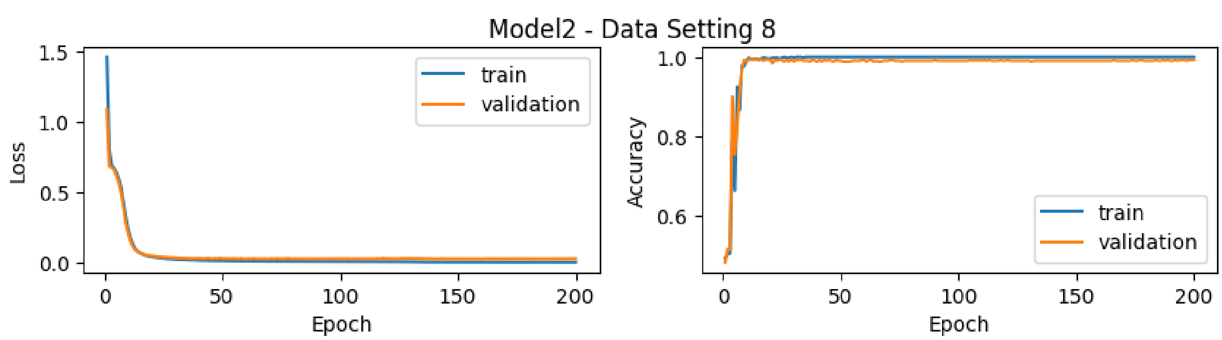

```
Experiment: 8 Setting: 5
N: 1000
```



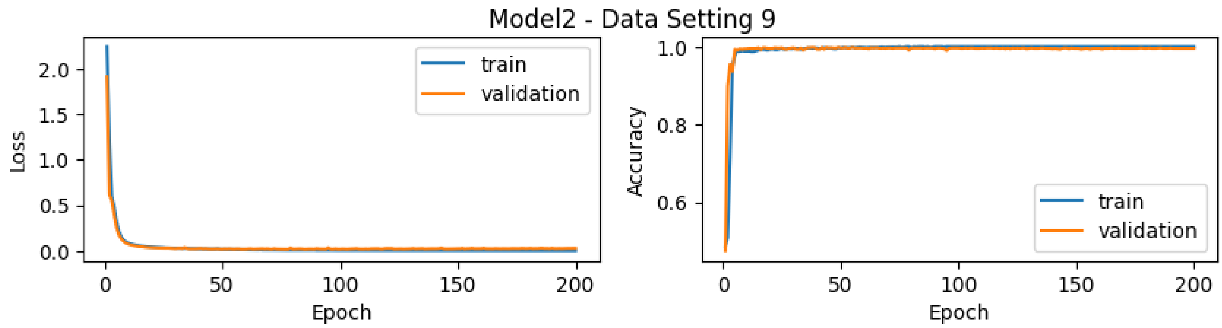

```
Experiment: 8 Setting: 6
N: 200
```



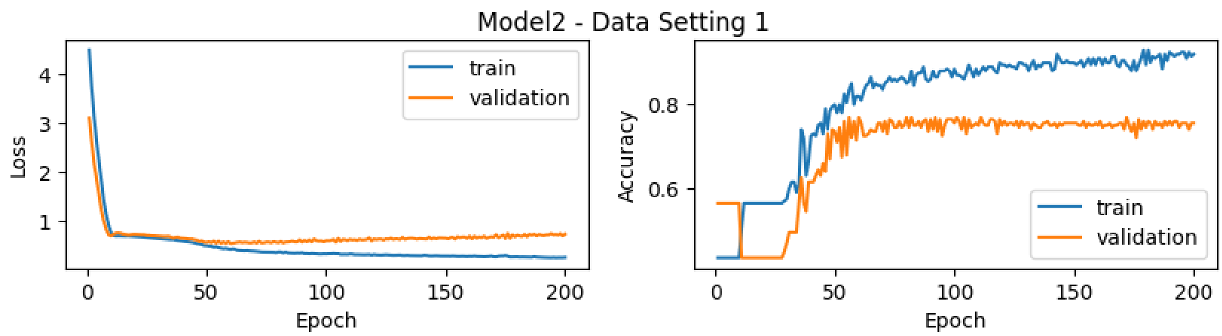

```
Experiment: 8 Setting: 7
N: 500
```



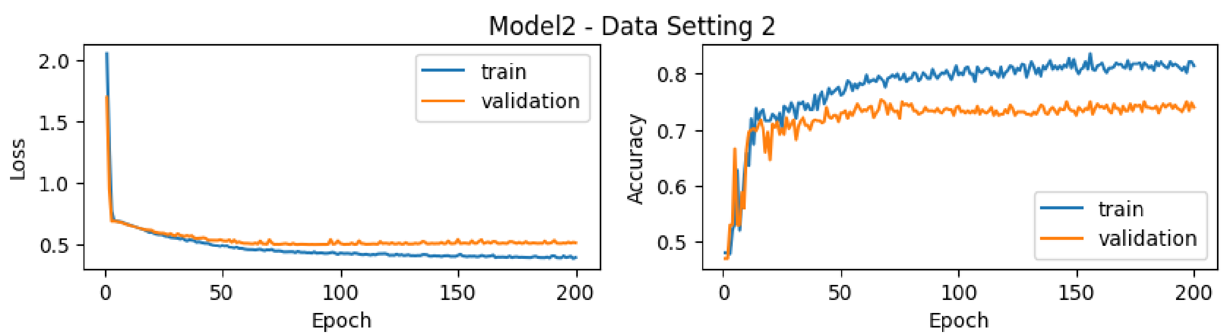

```
Experiment: 8 Setting: 8
N: 1000
```



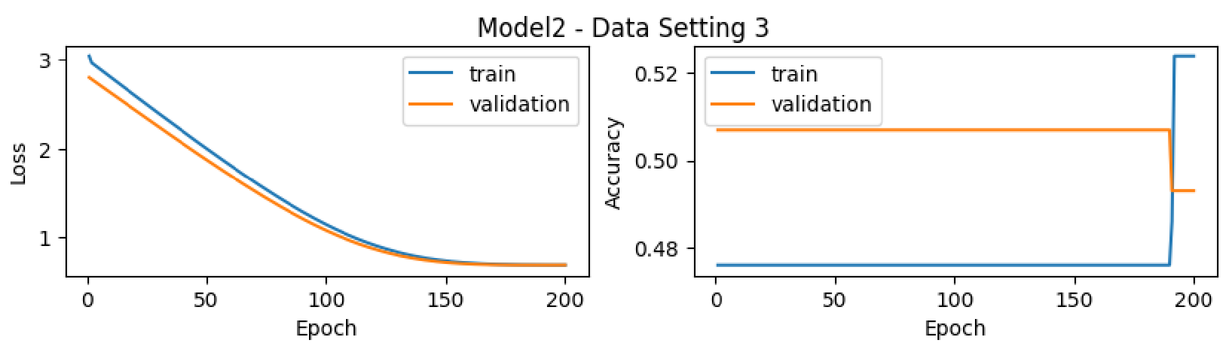

```
Experiment: 9 Setting: 0
N: 200
```



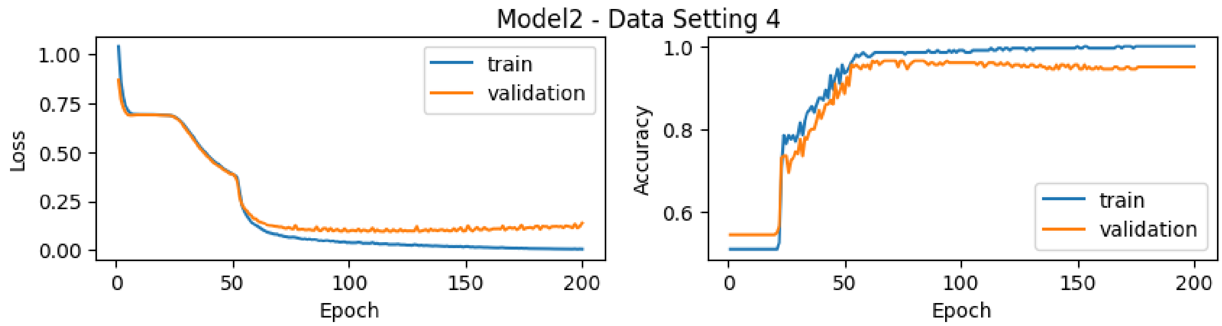

```
Experiment: 9 Setting: 1
N: 500
```



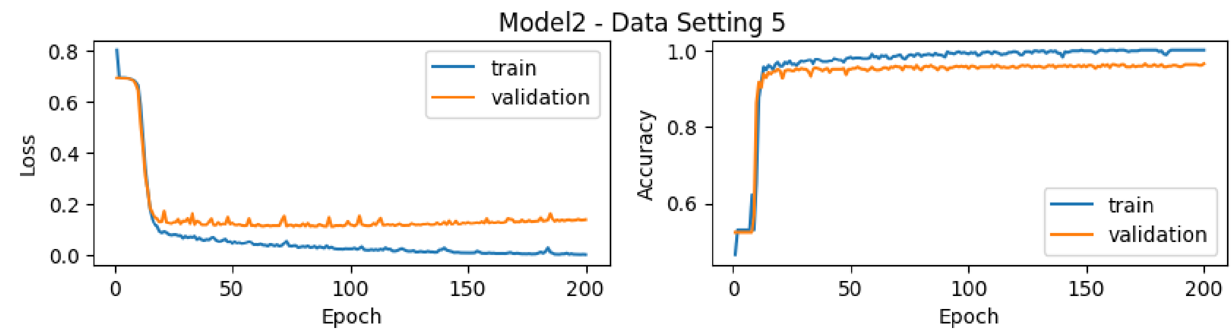

```
Experiment: 9 Setting: 2
N: 1000
```



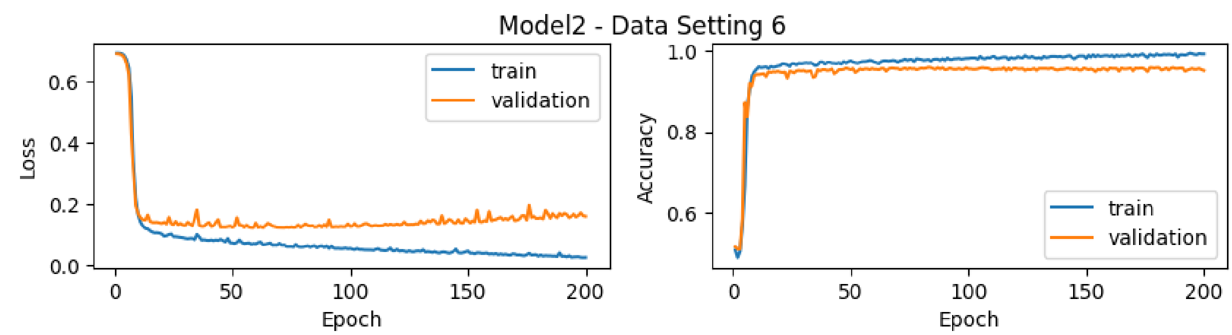

```
Experiment: 9 Setting: 3
N: 200
```



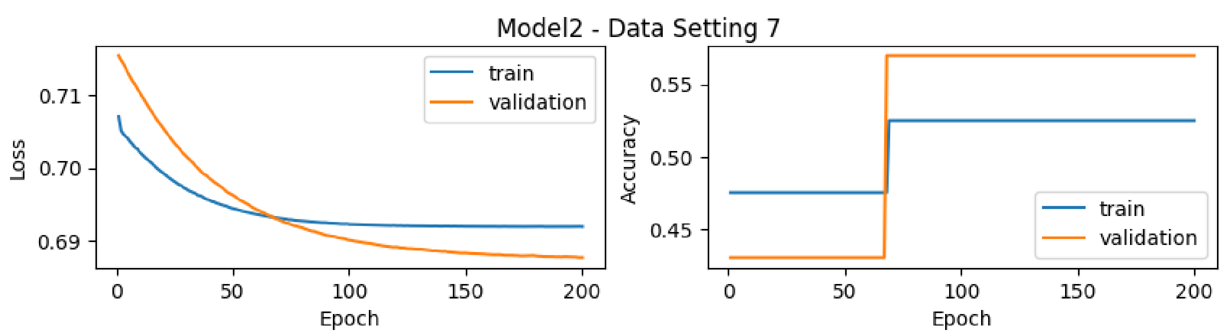

```
Experiment: 9 Setting: 4
N: 500
```



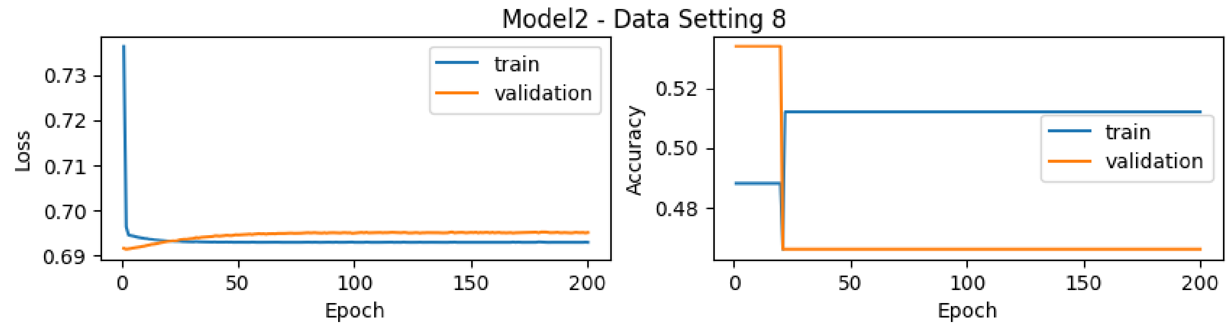

```
Experiment: 9 Setting: 5
N: 1000
```



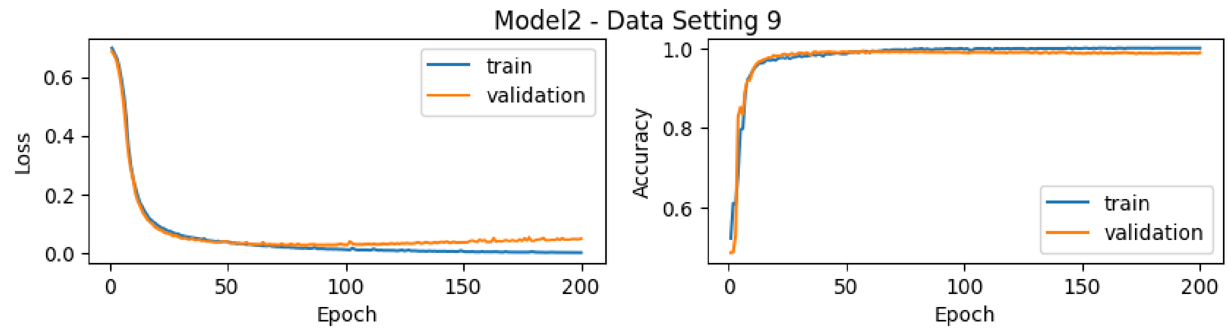

```
Experiment: 9 Setting: 6
N: 200
```




```
Experiment: 9 Setting: 7
N: 500
```




```
Experiment: 9 Setting: 8
N: 1000
```



In [ ]: