train(data, tokenMapping, gDff, gD\_k, gNLayers, gParallelHeads, gVocabSize, gBatchSize, gBlockSize, gPos, gEpochs, softmax, 'cuda')

**model\_20230821233603:**

data, gTokens, gVocabSize = getRawData()

gChannelsWanted = 128 # hopefully a good enough embdding to learn required features

gD\_k = gChannelsWanted

gDff = 256 #for intermediary feedforward process

gParallelHeads = 8 #for batch processing

gNLayers = 3 #how many blocks to use to make deeper network

gBatchSize = 5

gBlockSize = 10

gLearningRate = 1e-5

gPos = torch.arange(0, gBlockSize, dtype=torch.long) #to make a kind of positional embedding

gEpochs = 5

vocab = gTokens # !$&',-.3:;?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

tokenMapping = {k:v for v, k in enumerate(vocab)} #make dict where value is index of token and key is token itself

softmax = True

**model\_20230822000041:**

data, gTokens, gVocabSize = getRawData()

gChannelsWanted = 128 # hopefully a good enough embdding to learn required features

gD\_k = gChannelsWanted

gDff = 256 #for intermediary feedforward process

gParallelHeads = 8 #for batch processing

gNLayers = 4 #how many blocks to use to make deeper network

gBatchSize = 5

gBlockSize = 10

gLearningRate = 1e-5

gPos = torch.arange(0, gBlockSize, dtype=torch.long) #to make a kind of positional embedding

gEpochs = 5

vocab = gTokens # !$&',-.3:;?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

tokenMapping = {k:v for v, k in enumerate(vocab)} #make dict where value is index of token and key is token itself

softmax = True

**model\_20230822020436**

data, gTokens, gVocabSize = getRawData()

gChannelsWanted = 128 # hopefully a good enough embdding to learn required features

gD\_k = gChannelsWanted

gDff = 512 #for intermediary feedforward process

gParallelHeads = 8 #for batch processing

gNLayers = 4 #how many blocks to use to make deeper network

gBatchSize = 5

gBlockSize = 10

gLearningRate = 1e-5

gPos = torch.arange(0, gBlockSize, dtype=torch.long) #to make a kind of positional embedding

gEpochs = 5

vocab = gTokens # !$&',-.3:;?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

tokenMapping = {k:v for v, k in enumerate(vocab)} #make dict where value is index of token and key is token itself

softmax = True

**model\_20230822154709**

data, gTokens, gVocabSize = getRawData()

gChannelsWanted = 256 # hopefully a good enough embdding to learn required features

gD\_k = gChannelsWanted

gDff = 512 #for intermediary feedforward process

gParallelHeads = 8 #for batch processing

gNLayers = 5 #how many blocks to use to make deeper network

gBatchSize = 5

gBlockSize = 10

gLearningRate = 1e-5

gPos = torch.arange(0, gBlockSize, dtype=torch.long) #to make a kind of positional embedding

gEpochs = 5

vocab = gTokens # !$&',-.3:;?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

tokenMapping = {k:v for v, k in enumerate(vocab)} #make dict where value is index of token and key is token itself

softmax = True

**model\_20230823034208**

data, gTokens, gVocabSize = getRawData()

gChannelsWanted = 128 # hopefully a good enough embdding to learn required features

gD\_k = gChannelsWanted

gDff = 512 #for intermediary feedforward process

gParallelHeads = 4 #for batch processing

gNLayers = 4 #how many blocks to use to make deeper network

gBatchSize = 16

gBlockSize = 16

gLearningRate = 1e-4

gPos = torch.arange(0, gBlockSize, dtype=torch.long) #to make a kind of positional embedding

gEpochs = 20

vocab = gTokens # !$&',-.3:;?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

tokenMapping = {k:v for v, k in enumerate(vocab)} #make dict where value is index of token and key is token itself

softmax = False

**Model\_20230823041724**

data, gTokens, gVocabSize = getRawData()

gChannelsWanted = 64 # hopefully a good enough embdding to learn required features

gD\_k = gChannelsWanted

gDff = 512 #for intermediary feedforward process

gParallelHeads = 8 #for batch processing

gNLayers = 3 #how many blocks to use to make deeper network

gBatchSize = 16

gBlockSize = 16

gLearningRate = 1e-4

gPos = torch.arange(0, gBlockSize, dtype=torch.long) #to make a kind of positional embedding

gEpochs = 20

vocab = gTokens # !$&',-.3:;?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

tokenMapping = {k:v for v, k in enumerate(vocab)} #make dict where value is index of token and key is token itself

softmax = False

**model\_20230823193630**

data, gTokens, gVocabSize = getRawData()

gChannelsWanted = 64 # hopefully a good enough embdding to learn required features

gD\_k = gChannelsWanted

gDff = 256 #for intermediary feedforward process

gParallelHeads = 8 #for batch processing

gNLayers = 3 #how many blocks to use to make deeper network

gBatchSize = 16

gBlockSize = 16

gLearningRate = 1e-3

gPos = torch.arange(0, gBlockSize, dtype=torch.long) #to make a kind of positional embedding

gEpochs = 20

vocab = gTokens # !$&',-.3:;?ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

tokenMapping = {k:v for v, k in enumerate(vocab)} #make dict where value is index of token and key is token itself

softmax = False

**Model\_20230828032538**

**batchSize=8, blockSize=16, learningRate=0.001, epochs=20, parallelHeads=8, nLayers=3, channelsWanted=64, dff=128, softmax=False, usedProcessor='cuda'**