Table 1

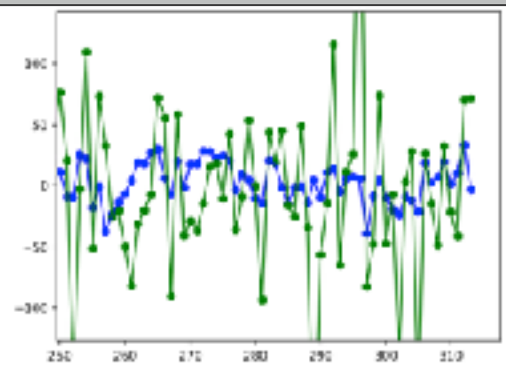
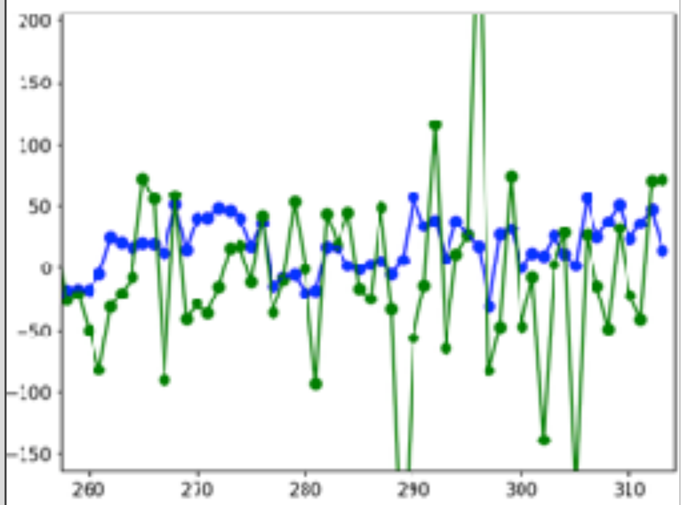
Model	Train jup notebook	Score(s)	Dataset
./state_dict15c.pt	c-15-predict-normalized-varience-not-randomness	**Valid:** 0.780438 **Test:** MSE loss: 0.57919404 MAE loss: 0.55981259 KLDiv loss: -0.06383617	**BATCH_SIZE** = 32 **SEQUENCE_SIZE** = 24 TARGETs_NAMEs = [] dataset=CryptoDataset(**predict_delta**=1, batch_size=BATCH_SIZE, sequence_size=SEQUENCE_SIZE)
./state_dict15c2.pt	c2-15-predict-normalized-varience-not-randomness	**Valid:** 0.786? **Test:** MSE loss: 0.62601354 MAE loss: 0.58926985 KLDiv loss: -0.09573120	**BATCH_SIZE** = 32 **SEQUENCE_SIZE** = 24 TARGETs_NAMEs = [] dataset=CryptoDataset(**predict_delta**=1, batch_size=BATCH_SIZE, sequence_size=SEQUENCE_SIZE)
./state_dict15c5.pt	c5-15-predict-normalized-varience-not-randomness-True	**Valid:** 0.635867 **Test:** MSE loss: 0.42487826 MAE loss: 0.52023978 KLDiv loss: -0.11454240	**BATCH_SIZE** = 32 **SEQUENCE_SIZE** = 24 TARGETs_NAMEs = [] dataset=CryptoDataset(**predict_delta**=1, batch_size=BATCH_SIZE, sequence_size=SEQUENCE_SIZE) **clipped**

Model	Model arch	Model forward

./state_dict15c.pt	SentimentNet(
	  (liniar): Linear(in_features=1829, out_features=3658, bias=True)
	  (normalization_pre_lstm): BatchNorm1d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
	  (lstm): LSTM(3658, 1829, num_layers=4, batch_first=True, dropout=0.4)
	  (fc): Sequential(
	  (0): Linear(in_features=1829, out_features=914, bias=True)
	  (1): BatchNorm1d(914, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
	  (2): LeakyReLU(negative_slope=0.01)
	  (3): Dropout(p=0.4, inplace=False)
	  (4): Linear(in_features=914, out_features=1, bias=True)
	  )
	)
		# Initialize hidden state with zeros
		  h0 = torch.zeros(self.n_layers, x.size(0), self.hidden_dim).requires_grad_().to(self.device)
		  # Initialize cell state
		  c0 = torch.zeros(self.n_layers, x.size(0), self.hidden_dim).requires_grad_().to(self.device)
		
		  out = self.liniar(x)
		  out = self.normalization_pre_lstm(out)
		  out = torch.tanh(out)
		  out, (hn, cn) = self.lstm(out, (h0, c0))
		  out = torch.add(out, x)
		  # Index hidden state of last time step
		  out = self.fc(out[:, -1, :])

./state_dict15c2.pt	SentimentNet(
	  (pre_lstm): Sequential(
	  (0): Linear(in_features=1829, out_features=3658, bias=True)
	  (1): BatchNorm1d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
	  (2): Tanh()
	  (3): Linear(in_features=3658, out_features=1829, bias=True)
	  (4): BatchNorm1d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
	  (5): Tanh()
	  )
	  (lstm): LSTM(1829, 1829, num_layers=4, batch_first=True, dropout=0.6)
	  (fc): Sequential(
	  (0): Linear(in_features=1829, out_features=914, bias=True)
	  (1): BatchNorm1d(914, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
	  (2): LeakyReLU(negative_slope=0.01)
	  (3): Dropout(p=0.6, inplace=False)
	  (4): Linear(in_features=914, out_features=1829, bias=True)
	  (5): BatchNorm1d(1829, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
	  (6): LeakyReLU(negative_slope=0.01)
	  (7): Dropout(p=0.6, inplace=False)
	  (8): Linear(in_features=1829, out_features=1, bias=True)
	  )
	)
		# Initialize hidden state with zeros
		  h0 = torch.zeros(self.n_layers, x.size(0), self.hidden_dim).requires_grad_().to(self.device)
		  # Initialize cell state
		  c0 = torch.zeros(self.n_layers, x.size(0), self.hidden_dim).requires_grad_().to(self.device)
		
		  out = self.pre_lstm(x)
		  lstm_out, (_, _) = self.lstm(out, (h0, c0))
		
		  # resblock
		  out = torch.add(lstm_out, out)
		
		  # Index hidden state of last time step
		  out = self.fc(out[:, -1, :])

./state_dict15c5.pt	SentimentNet(
	  (liniar): Linear(in_features=1829, out_features=3658, bias=True)
	  (normalization_pre_lstm): BatchNorm1d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
	  (lstm): LSTM(3658, 1829, num_layers=4, batch_first=True, dropout=0.4)
	  (fc): Sequential(
	  (0): Linear(in_features=1829, out_features=914, bias=True)
	  (1): BatchNorm1d(914, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
	  (2): LeakyReLU(negative_slope=0.01)
	  (3): Dropout(p=0.4, inplace=False)
	  (4): Linear(in_features=914, out_features=1, bias=True)
	  )
	)
		# Initialize hidden state with zeros
		  h0 = torch.zeros(self.n_layers, x.size(0), self.hidden_dim).requires_grad_().to(self.device)
		  # Initialize cell state
		  c0 = torch.zeros(self.n_layers, x.size(0), self.hidden_dim).requires_grad_().to(self.device)
		
		  out = self.liniar(x)
		  out = self.normalization_pre_lstm(out)
		  out = torch.tanh(out)
		  out, (hn, cn) = self.lstm(out, (h0, c0))
		  out = torch.add(out, x)
		  # Index hidden state of last time step
		  out = self.fc(out[:, -1, :])

Model	Test visualizer
./state_dict15c.pt	
./state_dict15c2.pt	
./state_dict15c5.pt