Better Deep Learning

Better Generalization vs Better Learning

STUDY CASES

* CARLOS FREDERICO CARVALHEIRA MELLO;
* CARMEM STEFANIE DA SILVA CAVALCANTE;
* EWERTON LEANDRO DE SOUSA.

Métodos

Better Generalization

- Force small weight with weight constraint;
- Halt training at the right time with early stopping.

Better Learning

- Fix vanishing gradiente with relu.

Constraint (restrição de pesos)

Extensões

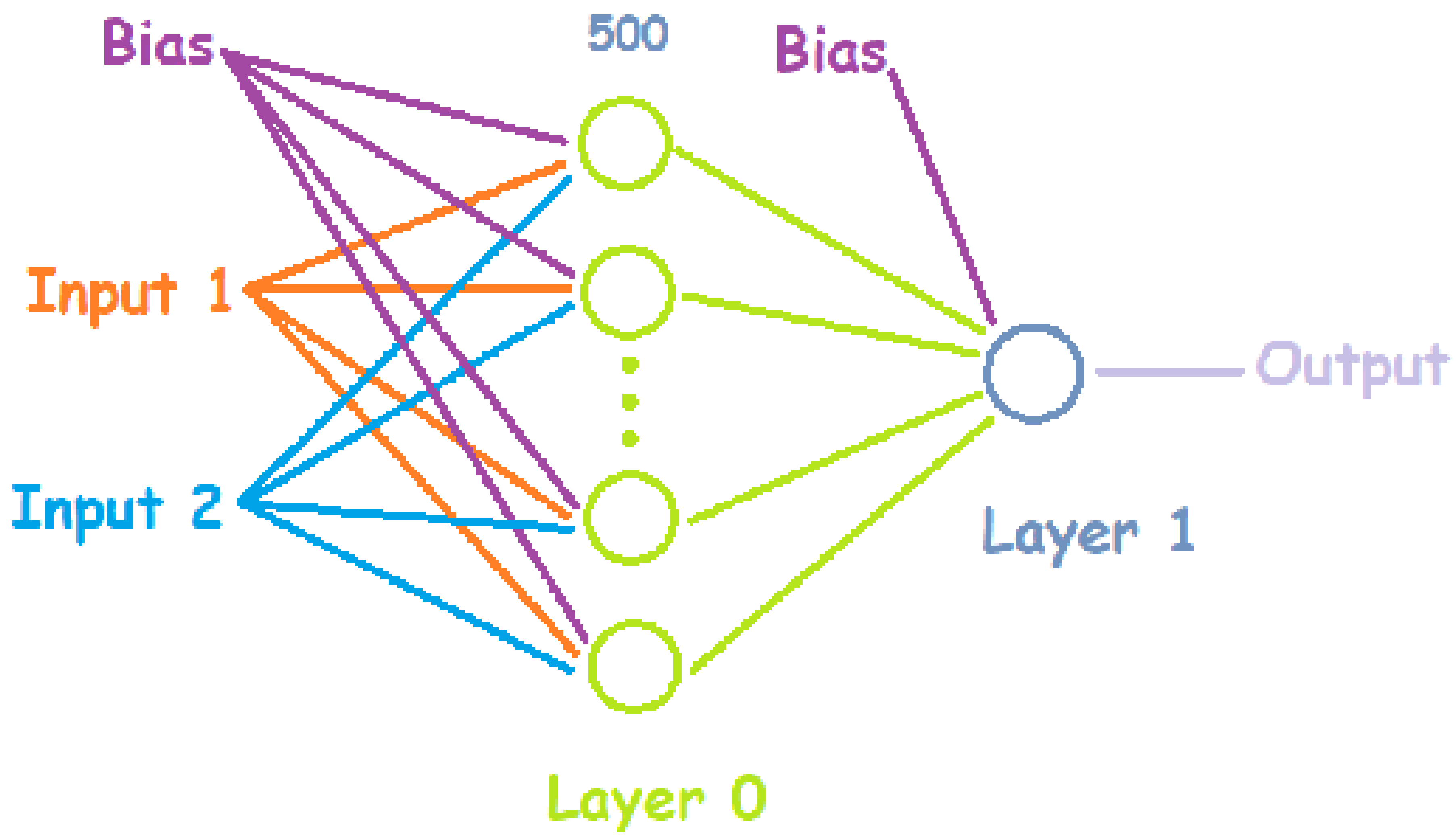
- Restrição de pesos em conjunto com regularização L2 ou Dropout;
- Demostrar a redução da magnitude dos pesos;
- Aplicar restrição de pesos na camada de saída;
- Aplicar restrição nas Bias.

REDE NEURAL

```
model_without = Sequential()
model_without.add(Dense(500, input_dim=2, activation='relu'))
model_without.add(Dense(1, activation='sigmoid'))
model_without.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

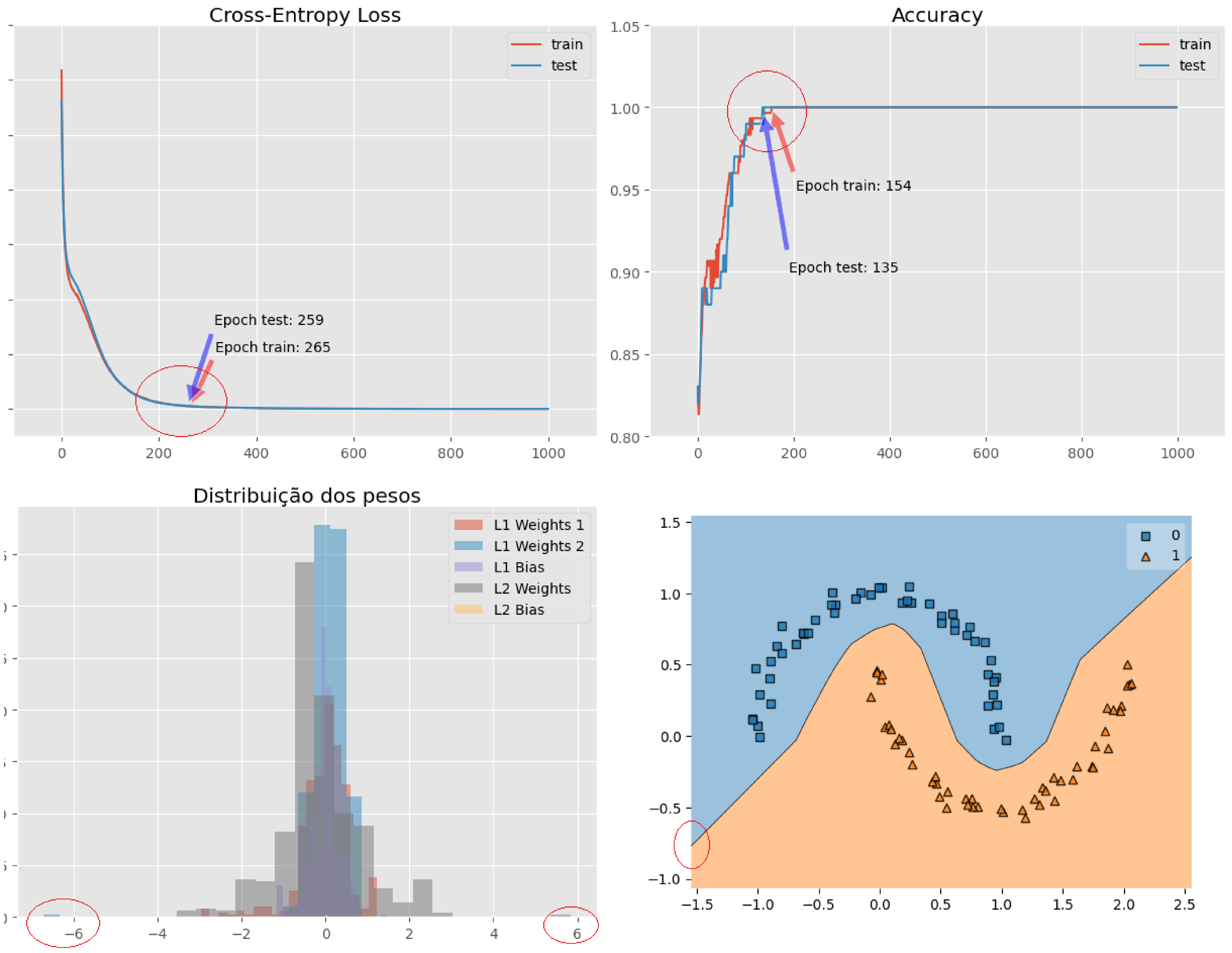
```
model_without.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 500)               1500
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 501
=================================================================
Total params: 2,001
Trainable params: 2,001
Non-trainable params: 0
_________________________________________________________________
```



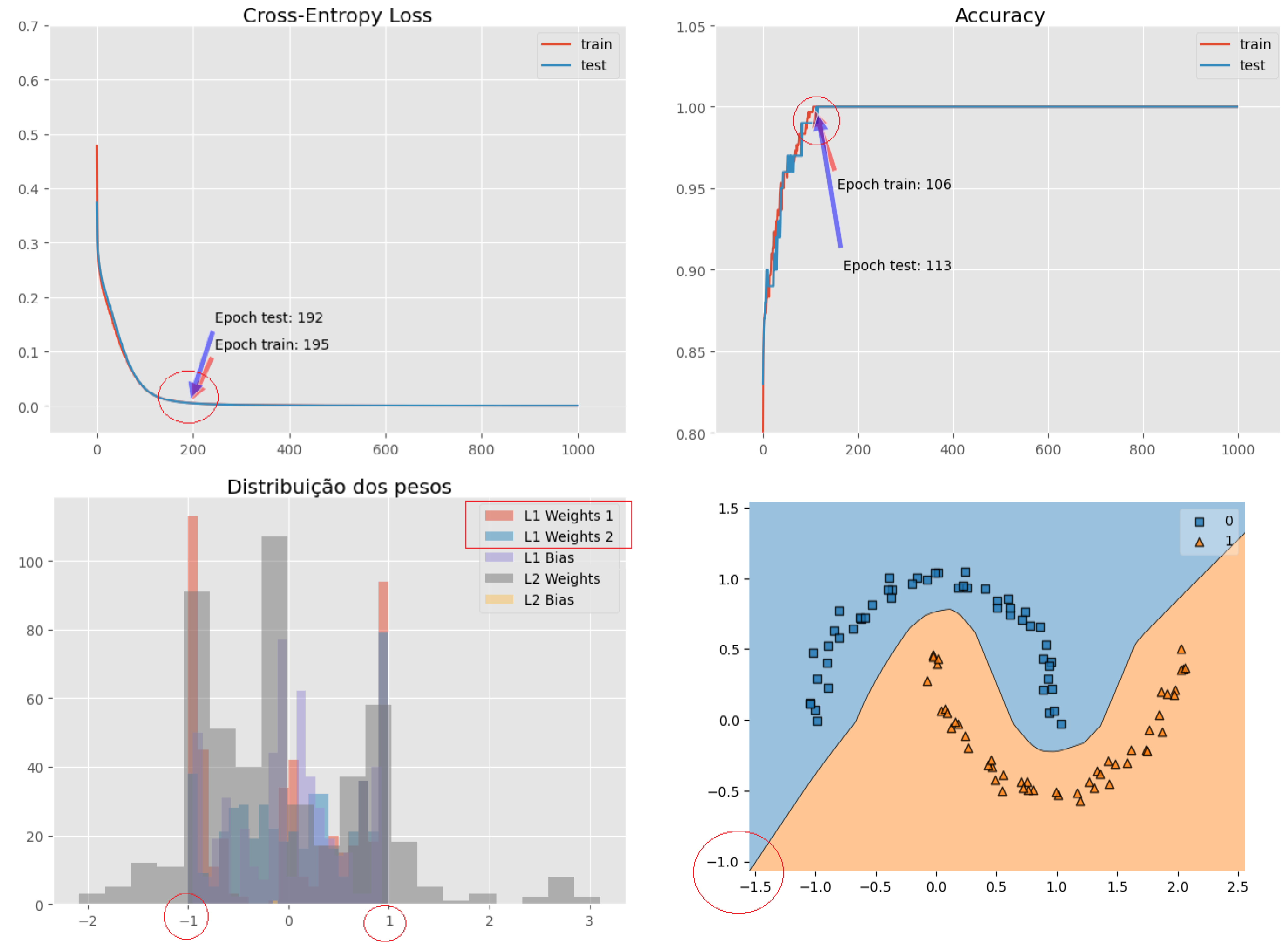
SEM RESTRIÇÃO

*model.fit(train_x, train_y, validation_data=(test_x, test_y), **epochs=4000**, verbose=0, callbacks=[MyCustomCallback(),tensorboard_callback])*

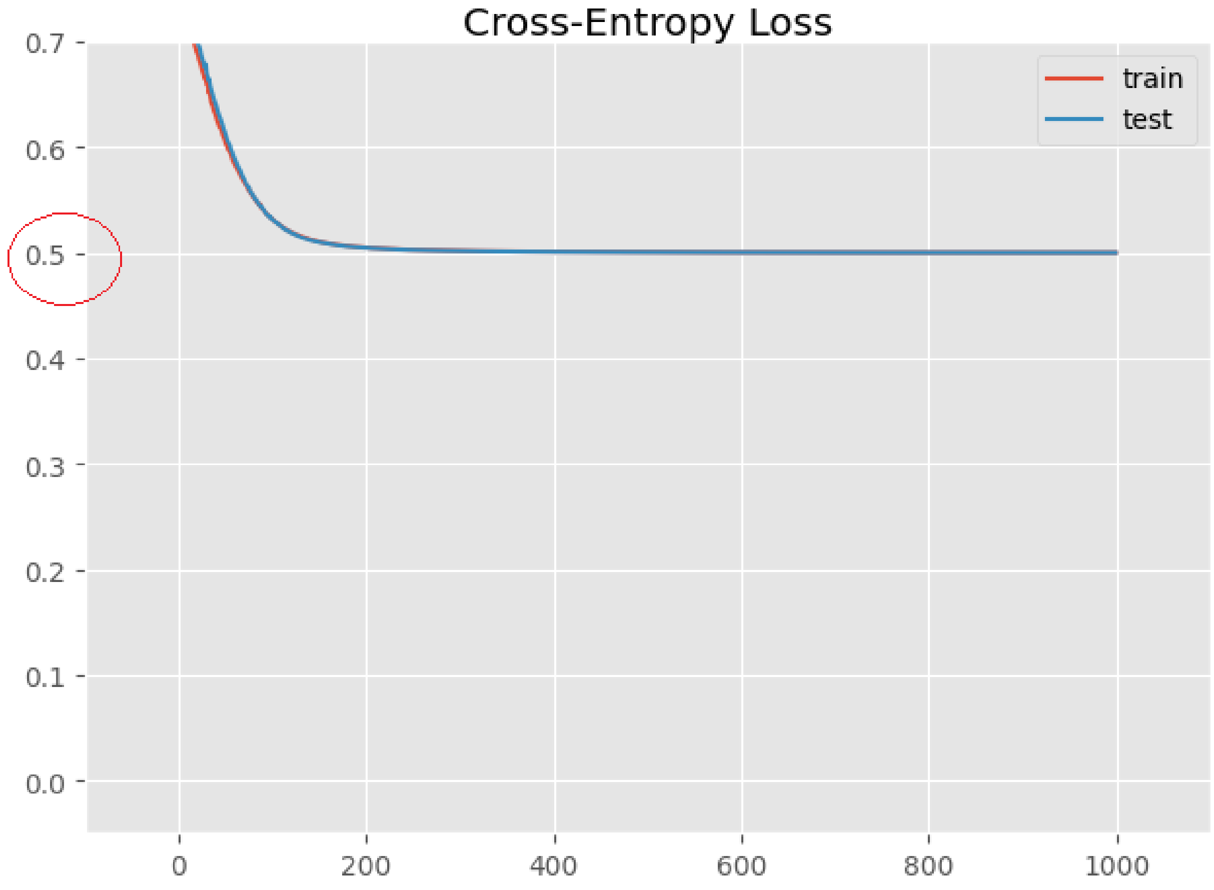


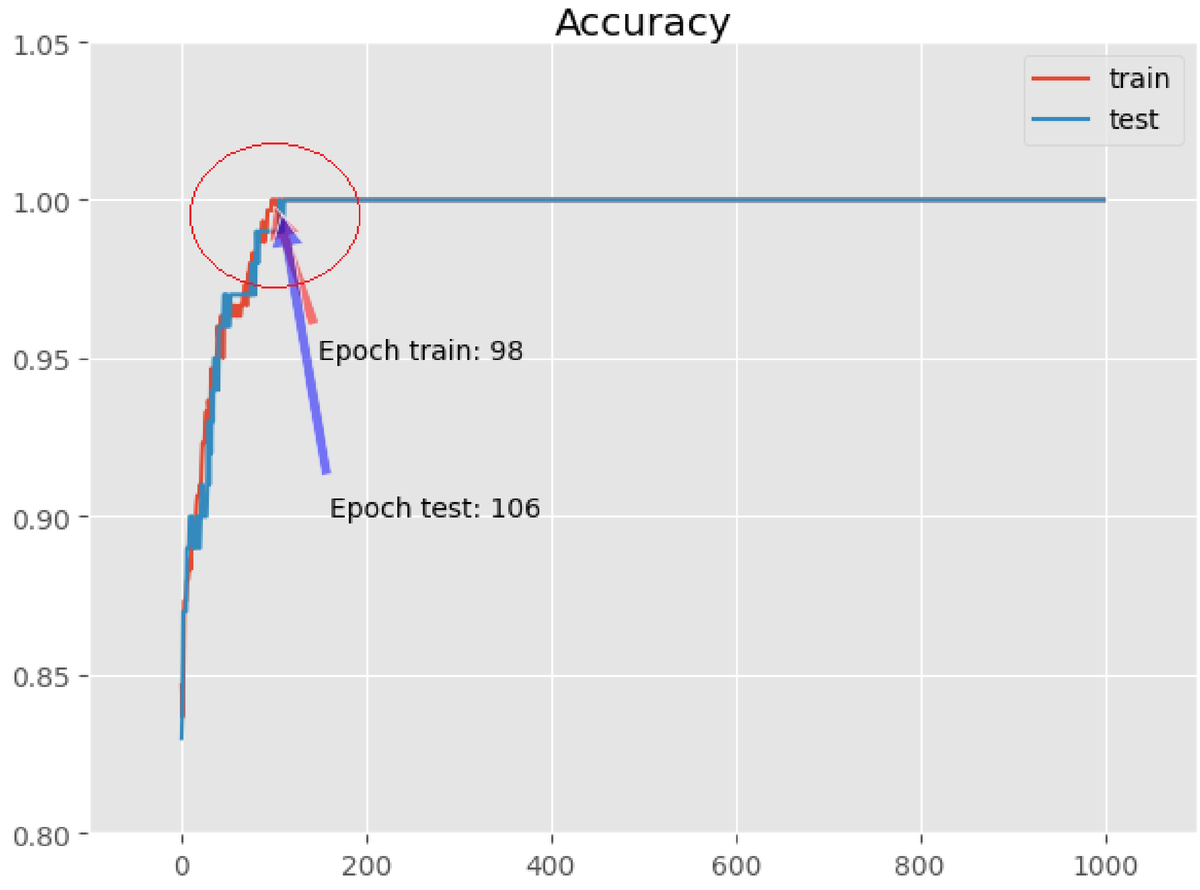
COM RESTRIÇÃO (DISTRIBUIÇÃO NORMAL)

model.add(Dense(500, input_dim=2, activation='relu', kernel_constraint=unit_norm()))

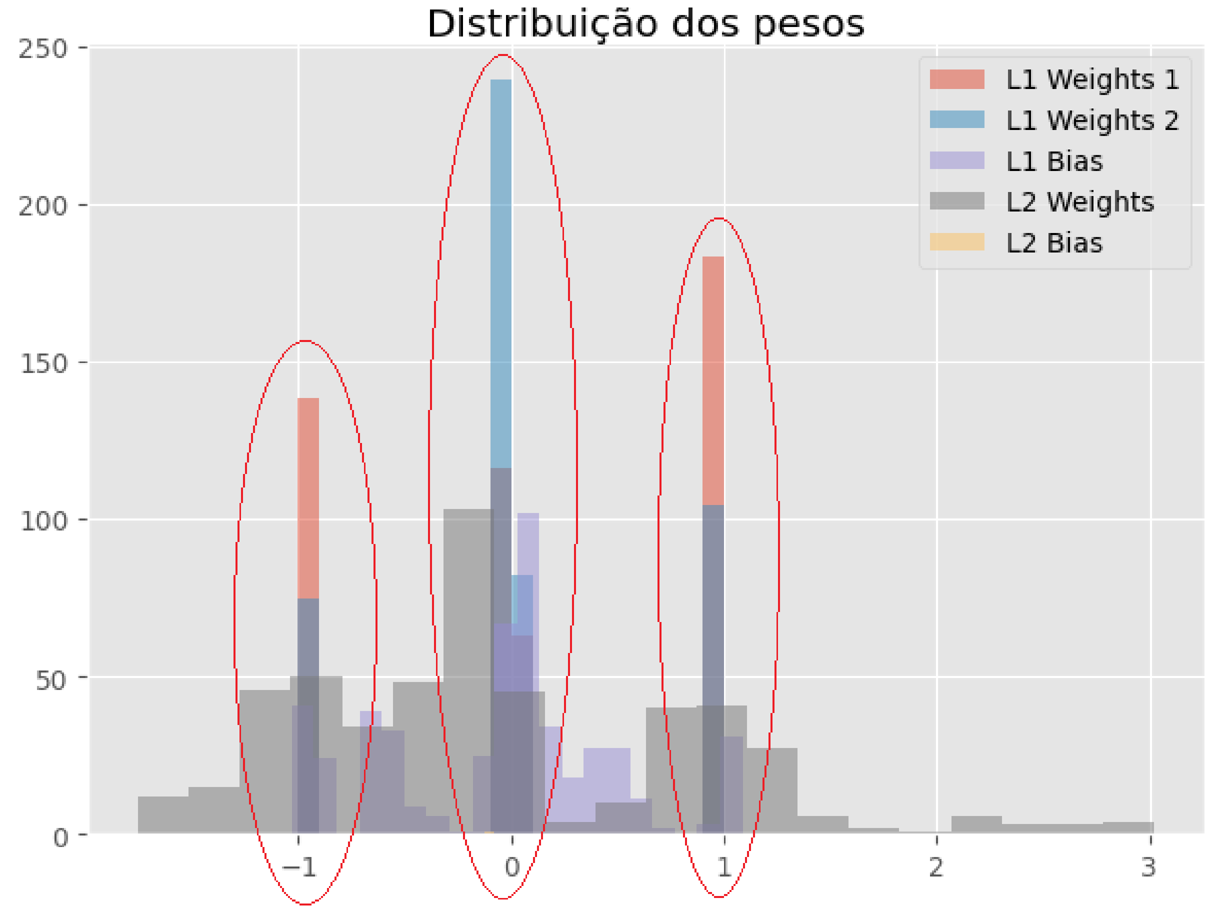


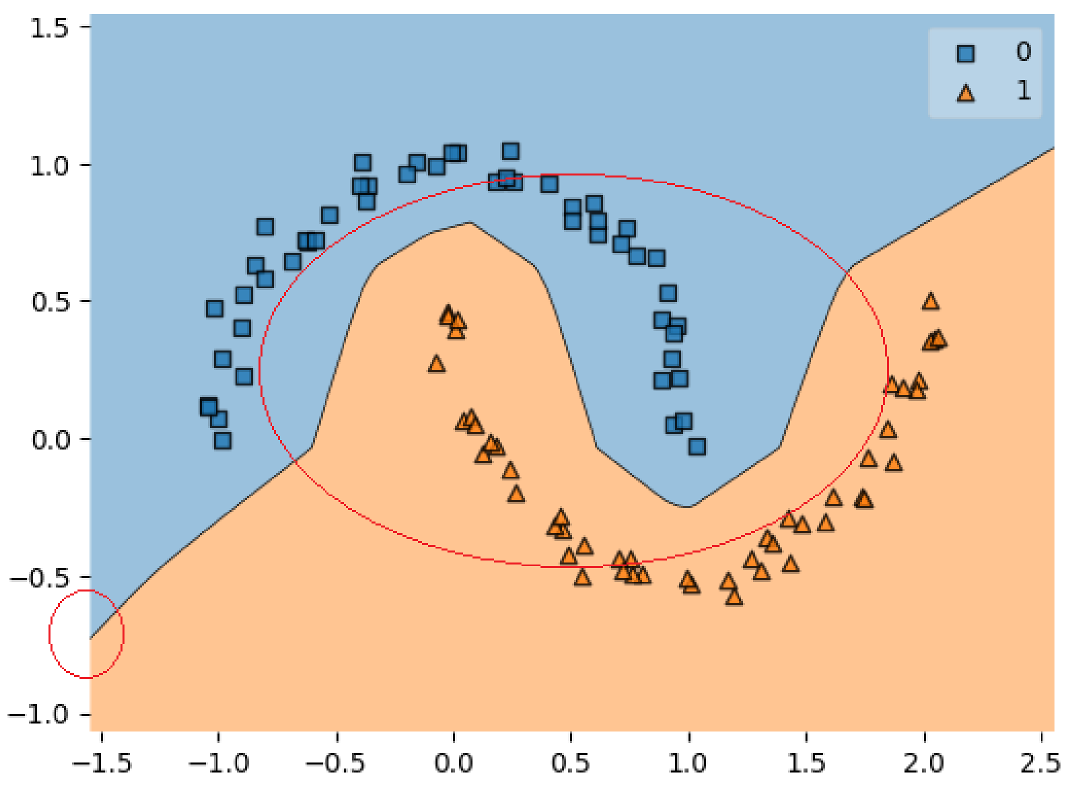
COM RESTRIÇÃO E REGULARIZAÇÃO L2





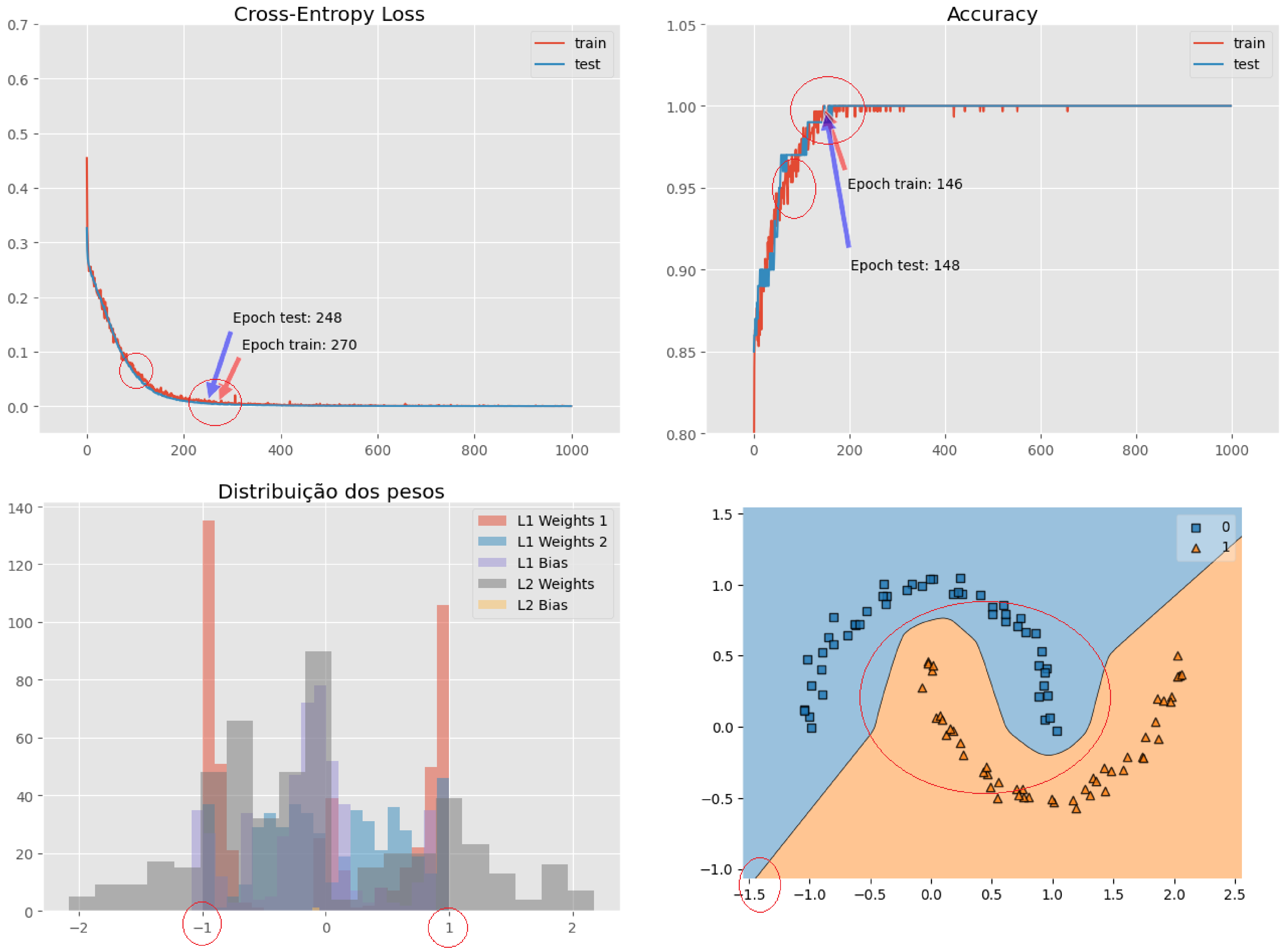
Não foi observado melhora na generalização a utilização da restrição do peso com regularização L2.





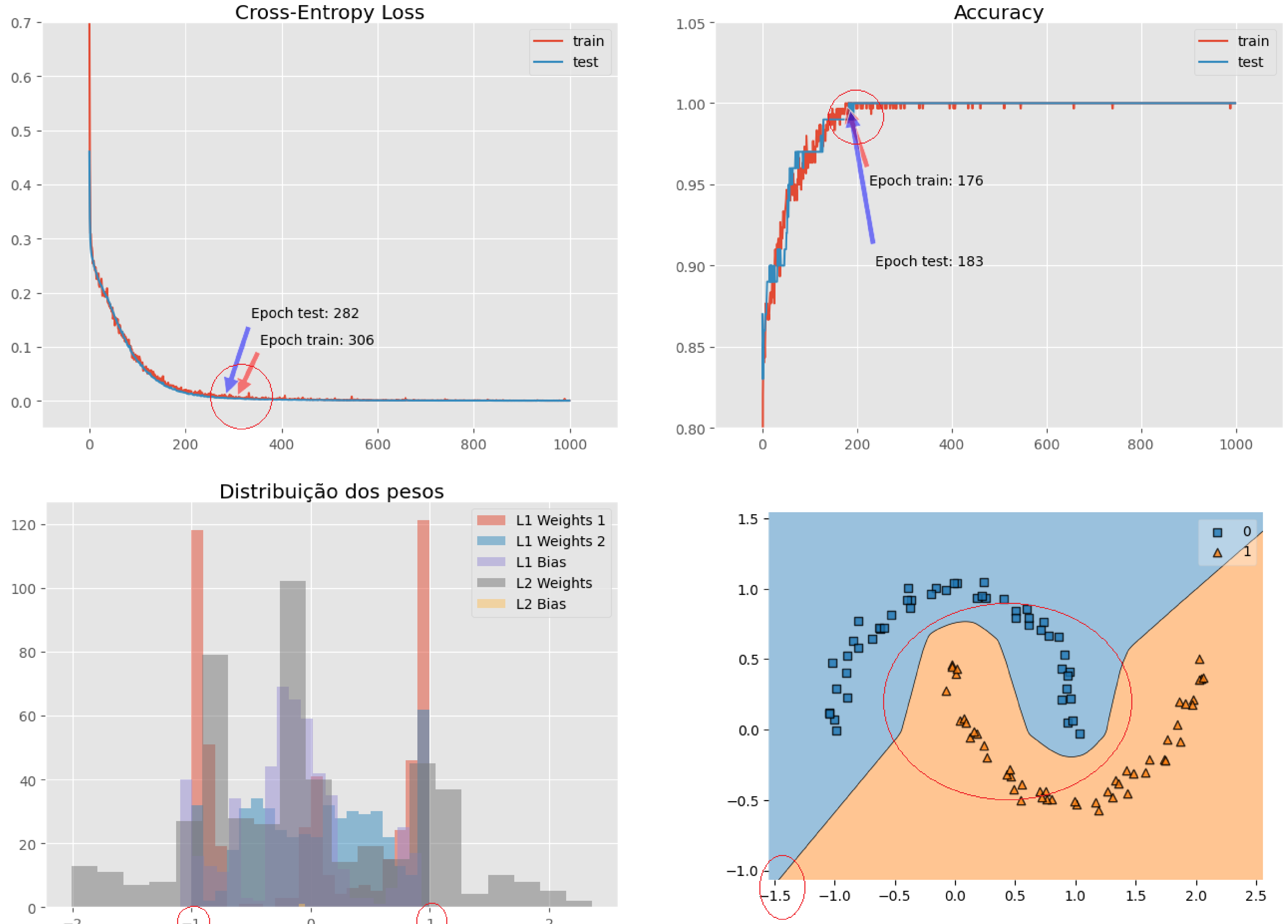
COM RESTRIÇÃO E DROPOUT(0.4)

Foi observado melhora na generalização com a utilização da restrição do peso com dropout.



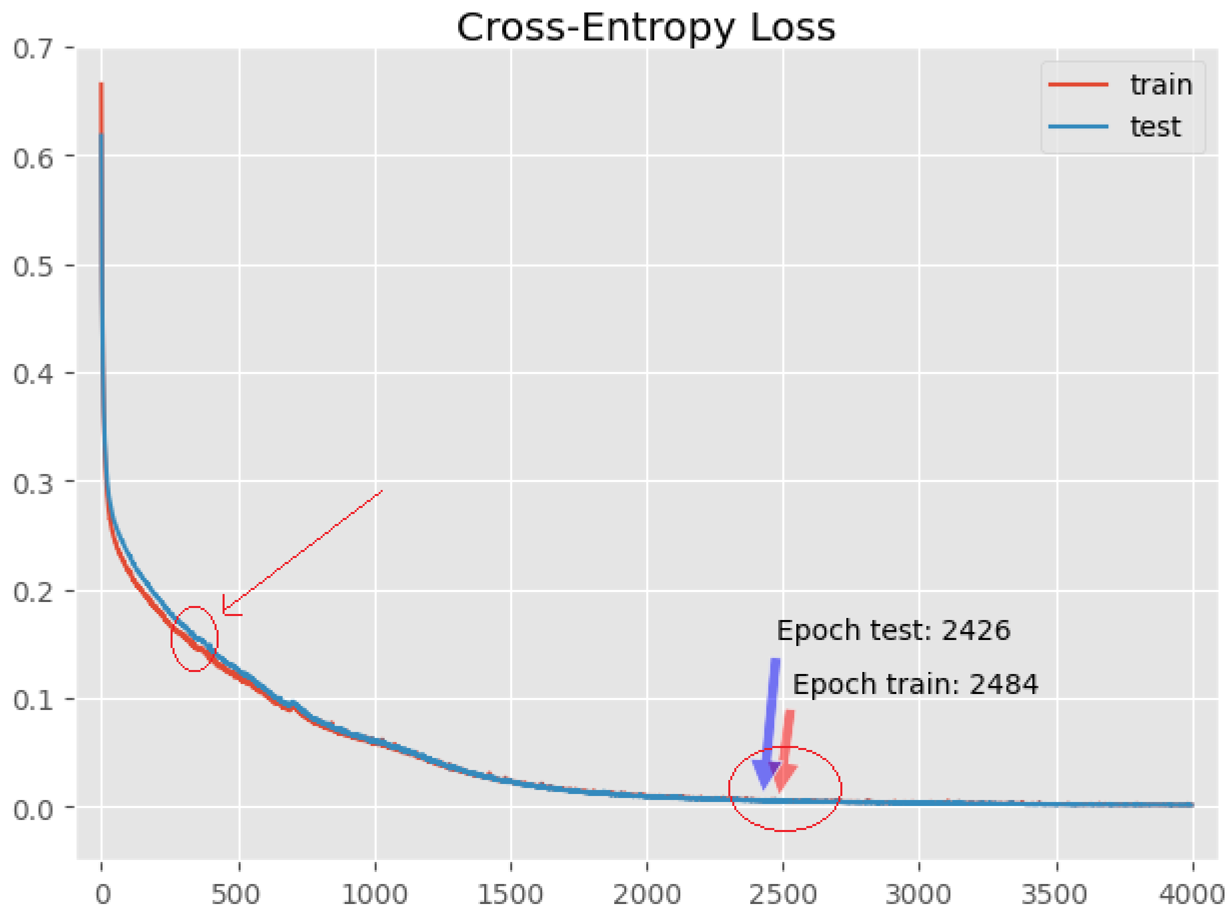
COM RESTRIÇÃO, REGULARIZAÇÃO L2 E DROPOUT(0.4)

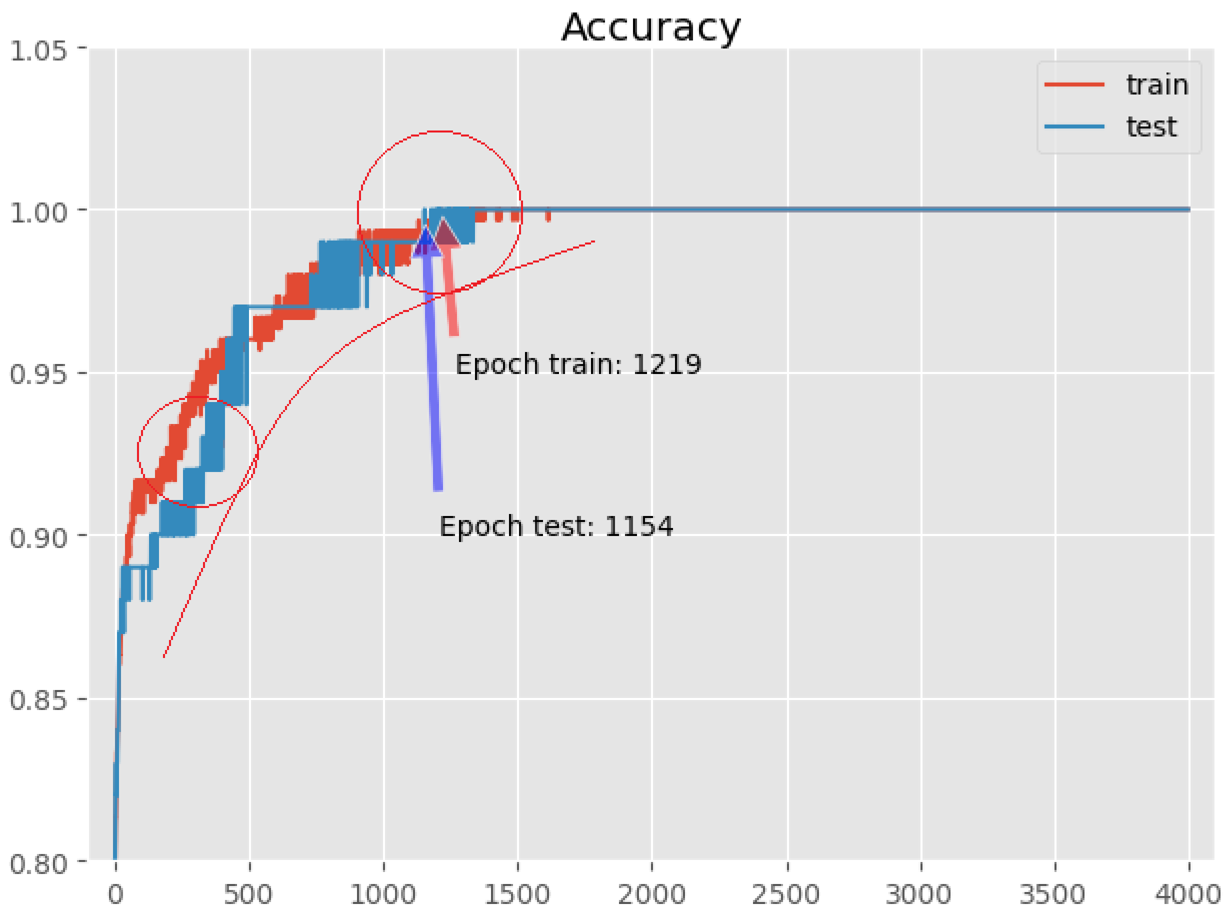
A combinação dos 3 métodos foi o que obteve o melhor resultado.

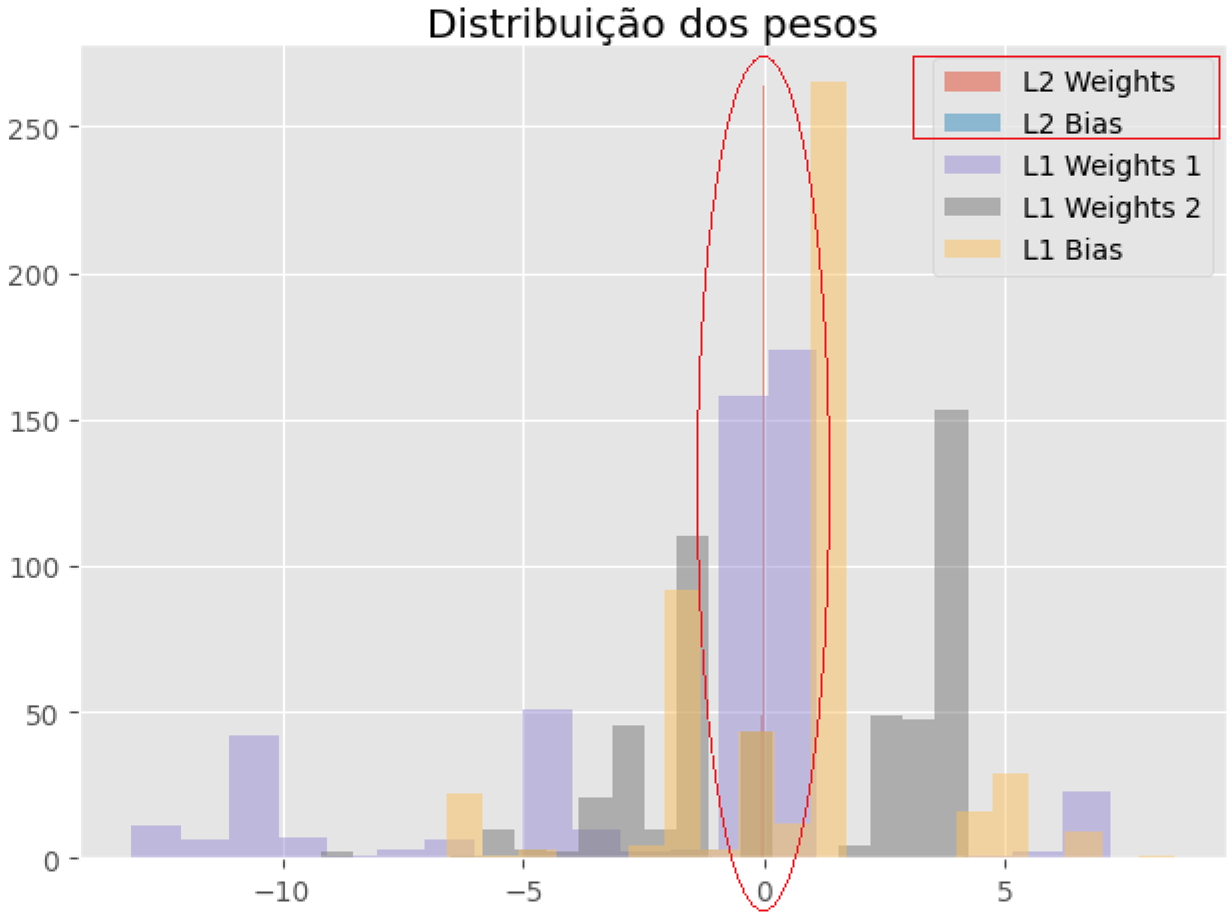


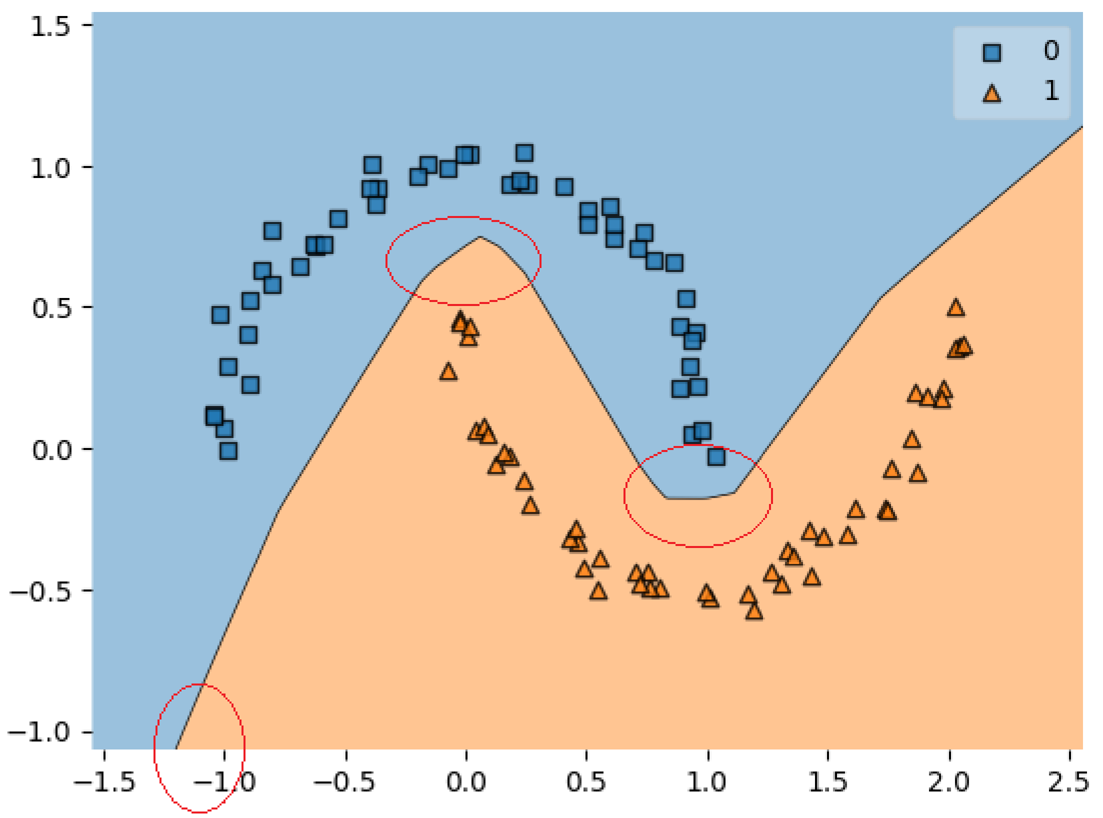
COM RESTRIÇÃO NA CAMADA DE SAÍDA

model.add(Dense(1, activation='sigmoid', kernel_constraint=unit_norm()))



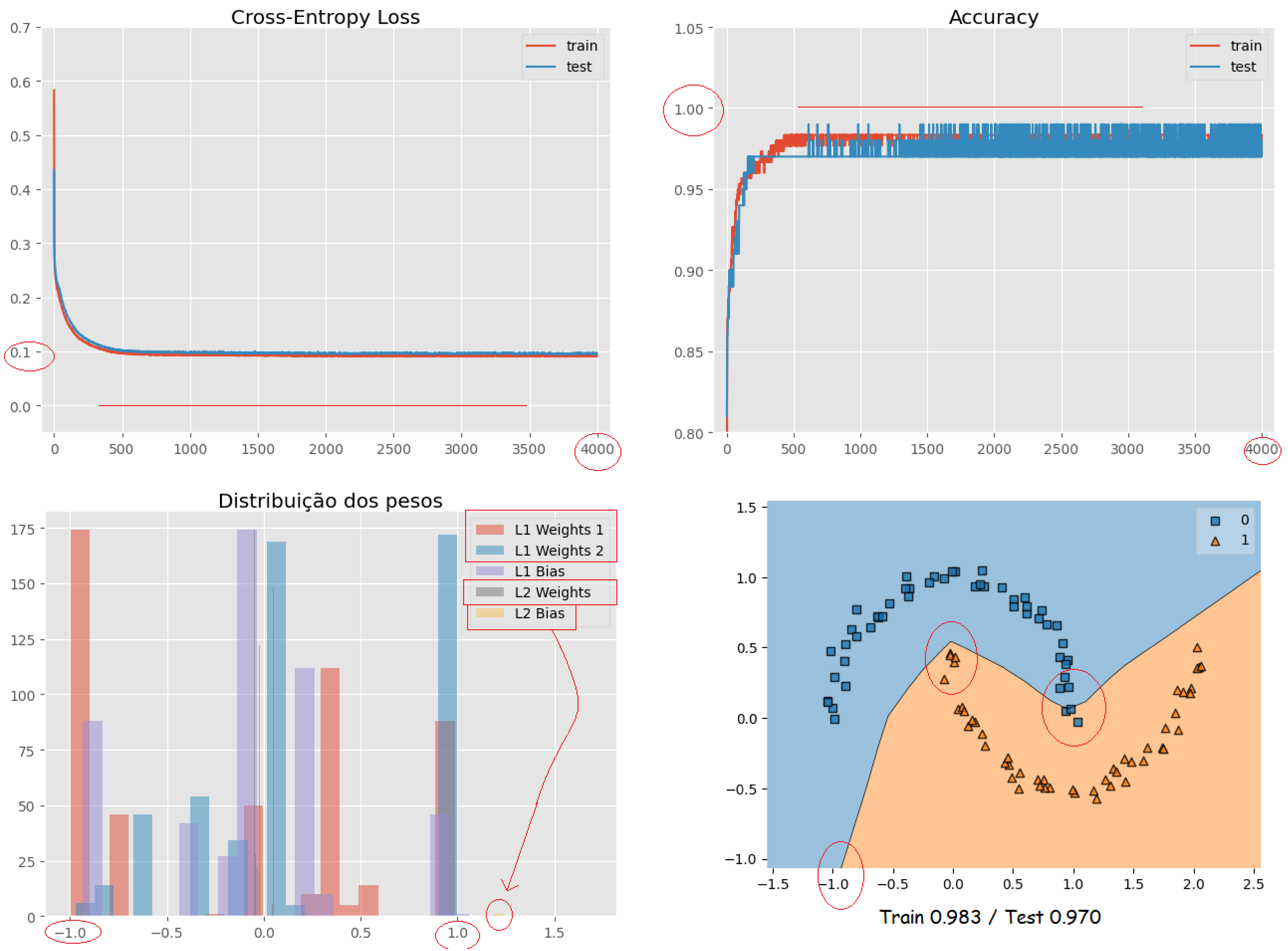






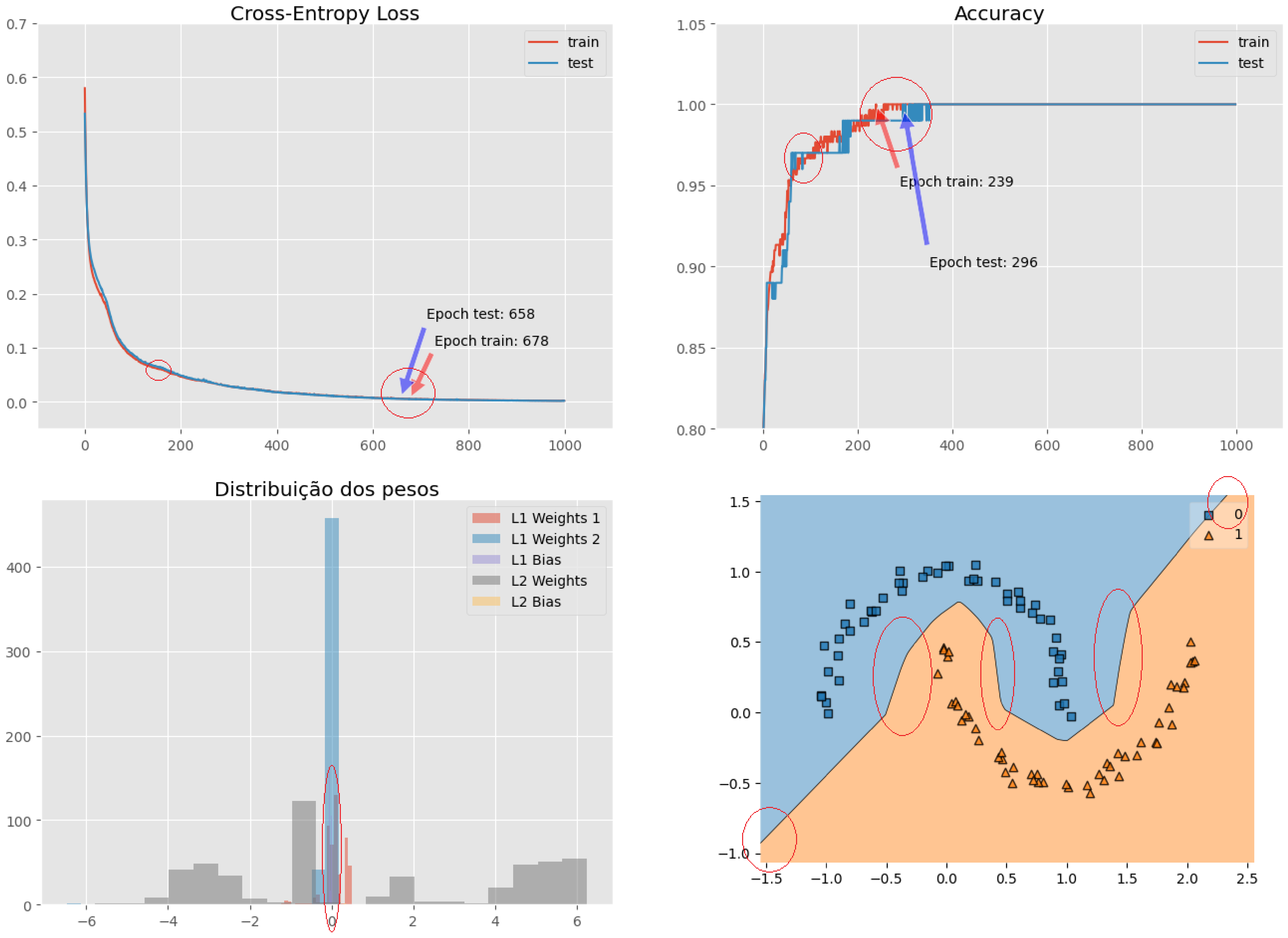
COM RESTRIÇÃO NA CAMADA DE ENTRADA E SAÍDA

Único caso de estudo em que o modelo não conseguiu ter um erro igual a 0 e 100% de acurácia.



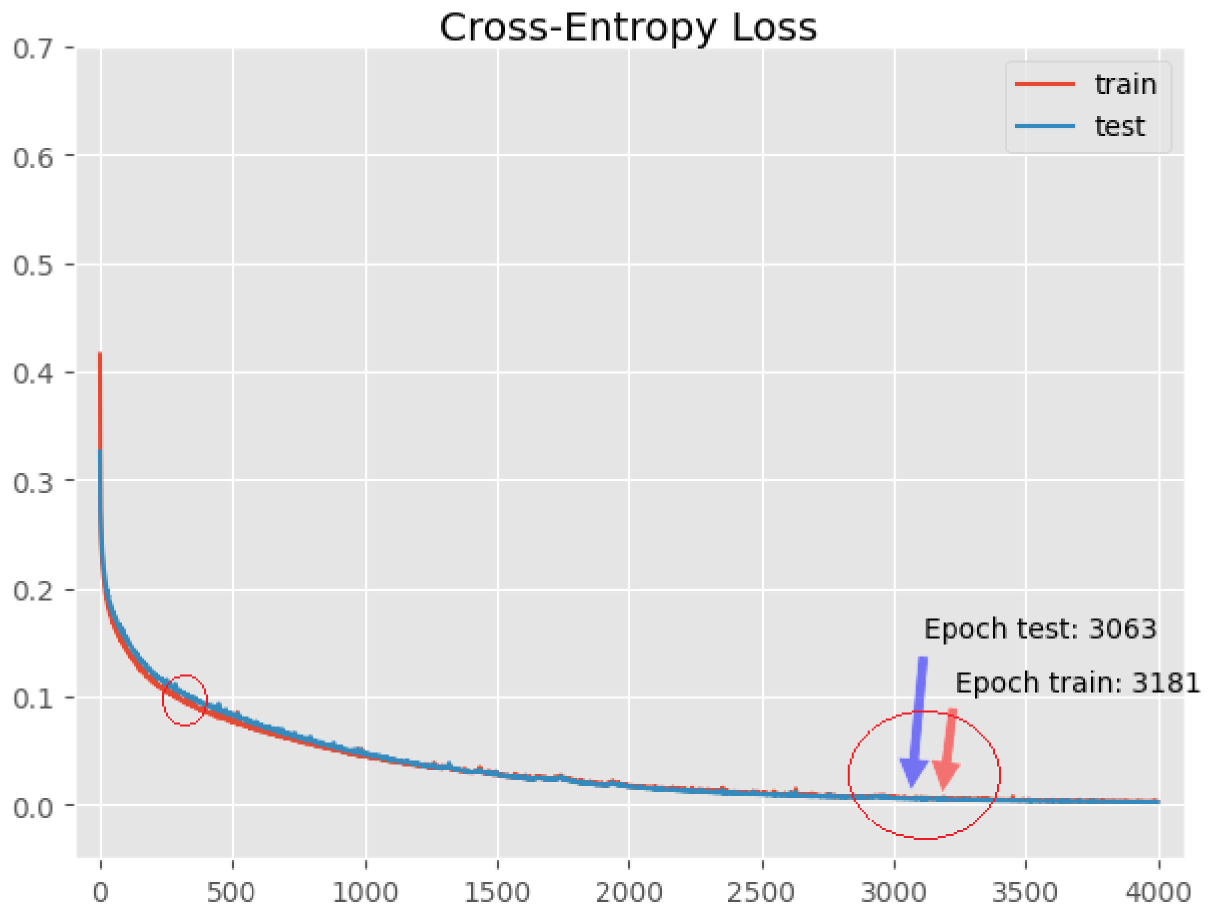
COM RESTRIÇÃO NA BIAS

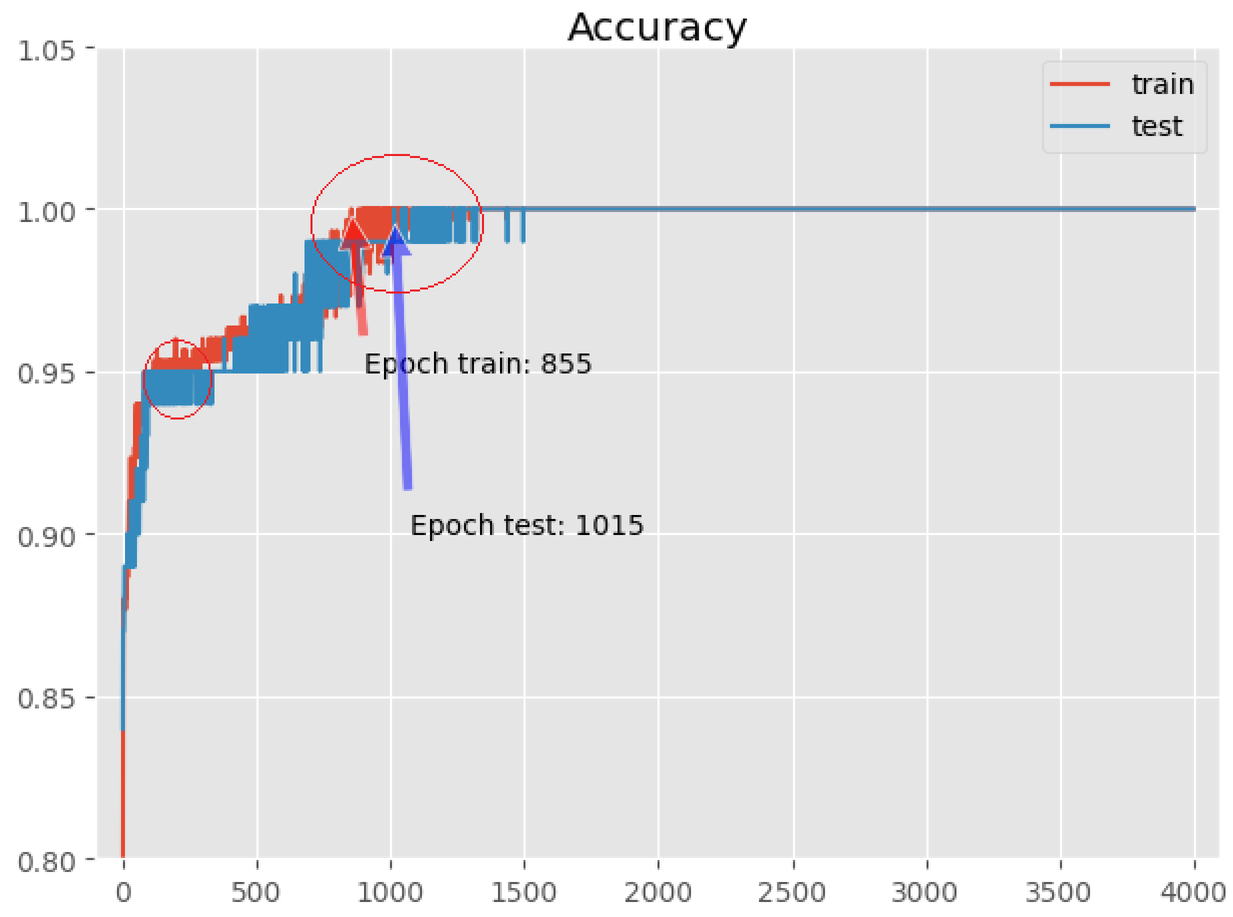
*model.add(Dense(500, input_dim=2, activation='relu', **bias_constraint**=unit_norm()))*

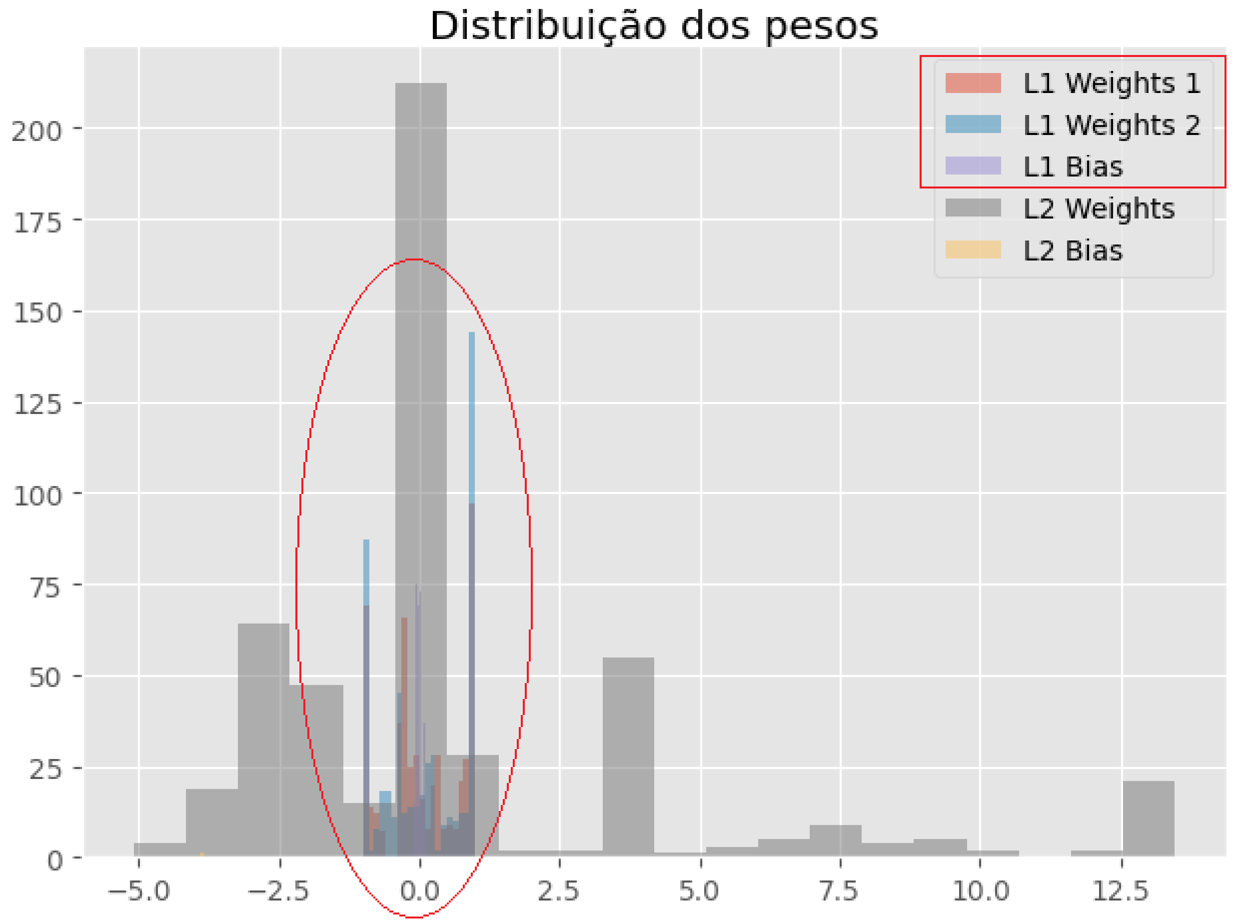


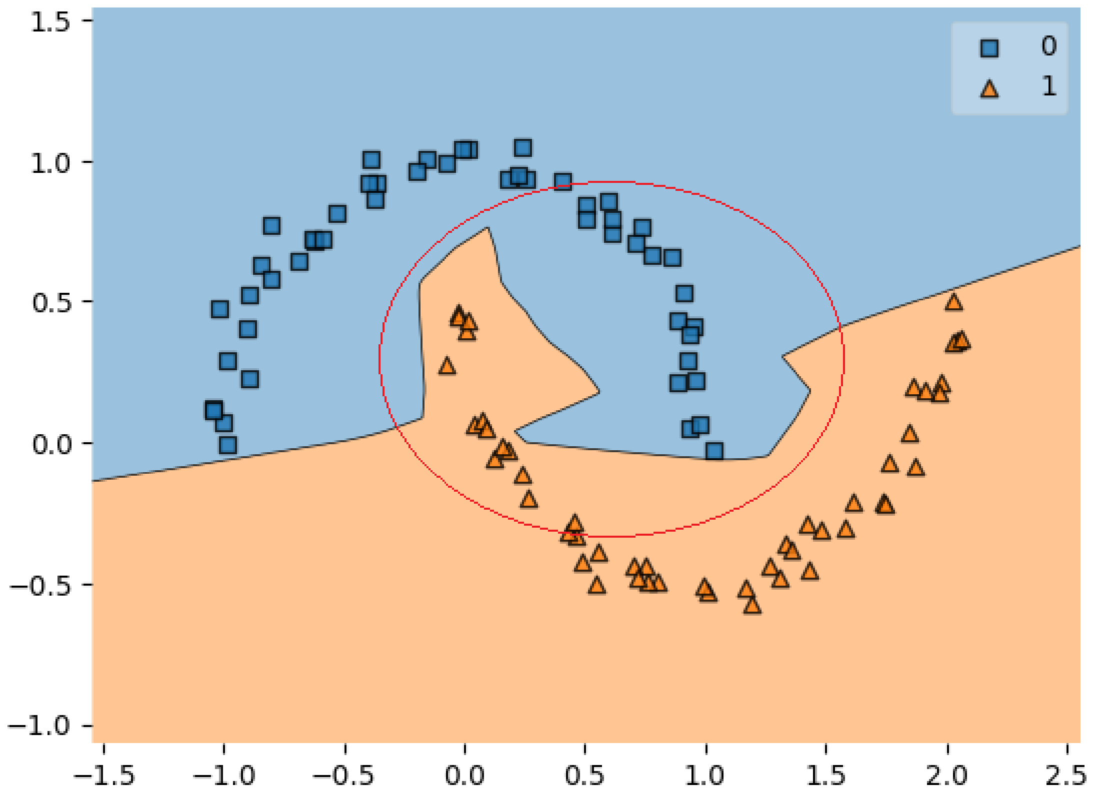
COM RESTRIÇÃO NOS PESOS E NAS BIAS

*model.add(Dense(500, input_dim=2, activation='relu', **kernel_constraint**=unit_norm() , **bias_constraint**=unit_norm()))*









CONCLUSÕES

A utilização de restrições

teve um impacto positivo forçando a rede a generalização, porém a utilização de muitos métodos de regularização em conjuntos ou em varias camadas da rede pode causa problema de underfitting.

Early Stopping

Extensões

- Usando acurácia
- Usando dados de validação
- Em um problema de regressão

MONITORANDO A ACURÁCIA

ES monitorando loss

Servir de base comparativa para as execuções posteriores.

ES simples

Na primeira vez que a época seguinte já apresentar piora em relação a atual, para o modelo.

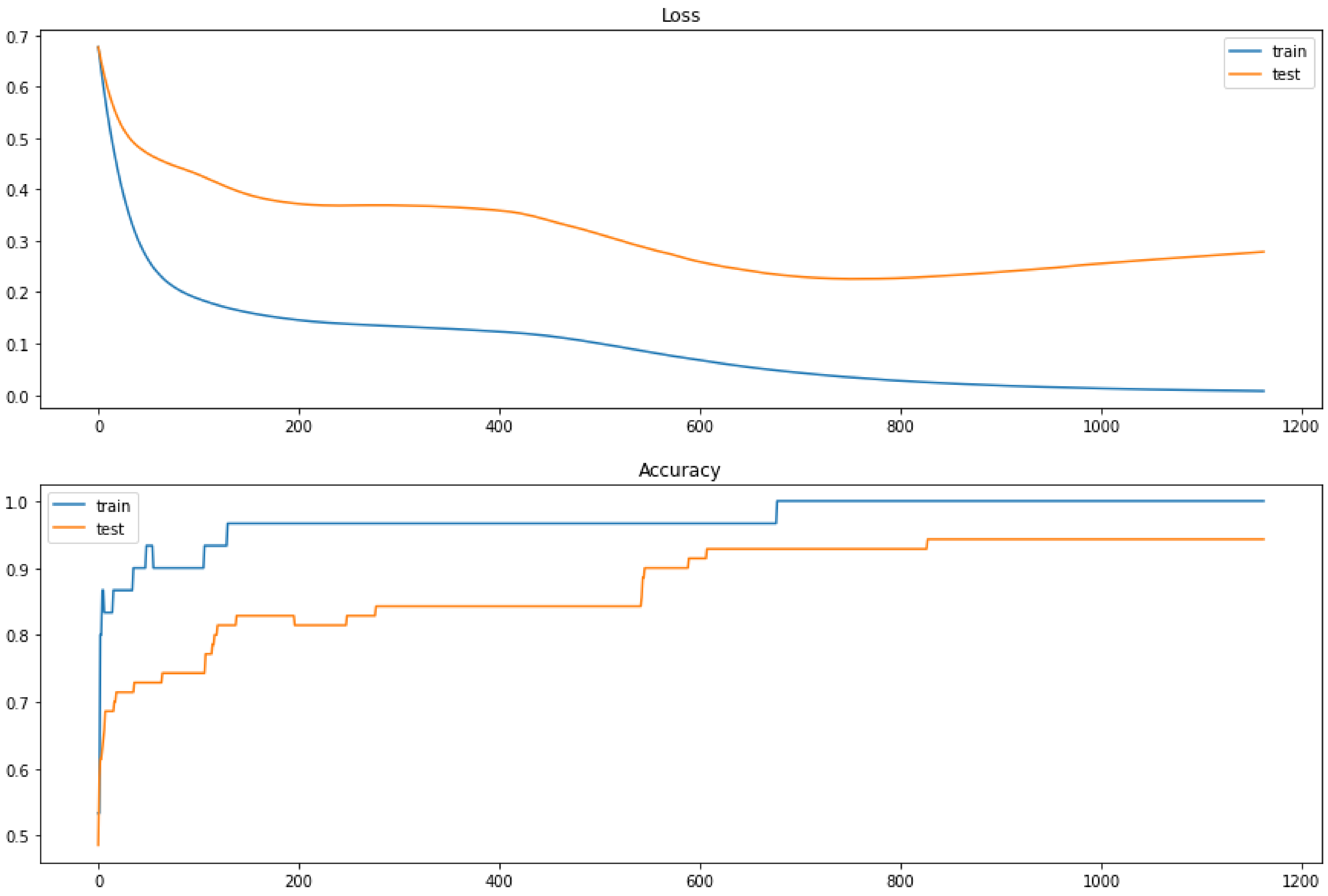
ES com patience

Espera um determinado número de épocas, após encontrar uma piora na variável que está sendo monitorada, antes de parar o modelo.

ES e Modelcheckpoint

Durante a execução, guarda o melhor modelo em um checkpoint separado, que é acessado ao fim.

MONITORANDO LOSS

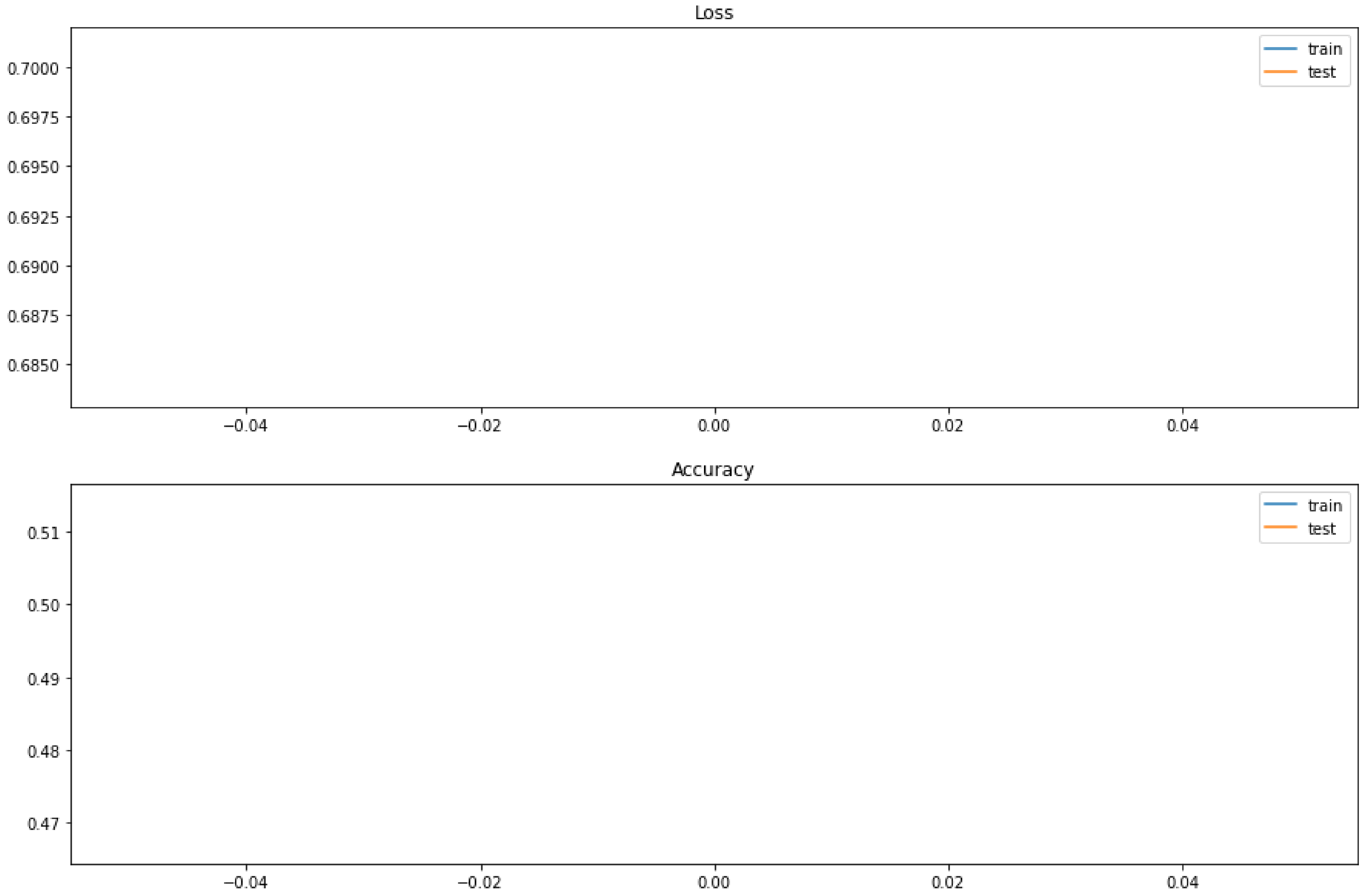


Epoch 01163: early stopping
Train: 1.000, Test: 0.943

ES SIMPLES

Epoch 1/4000

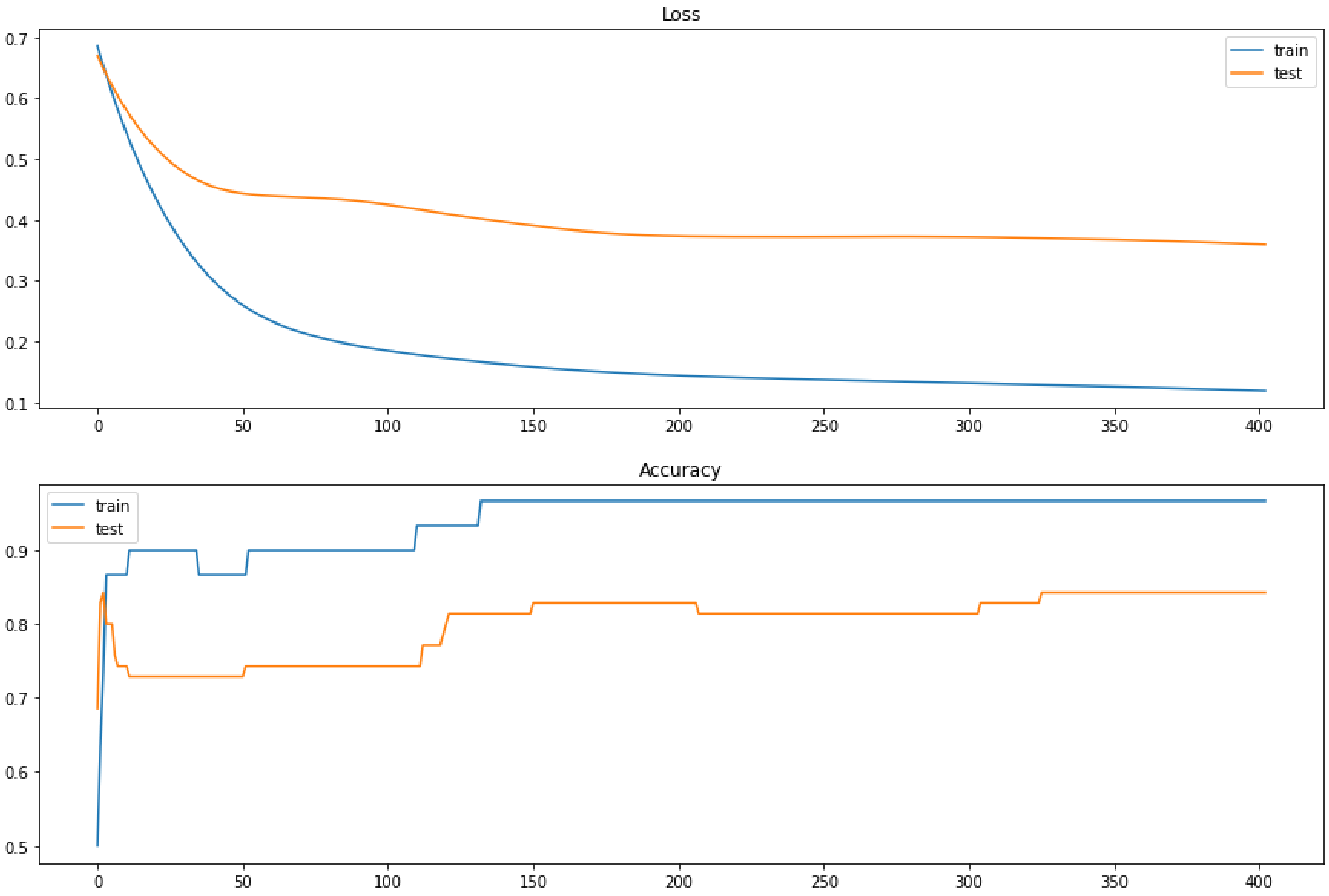
Train: 0.467, Test: 0.514



ES + PATIENCE

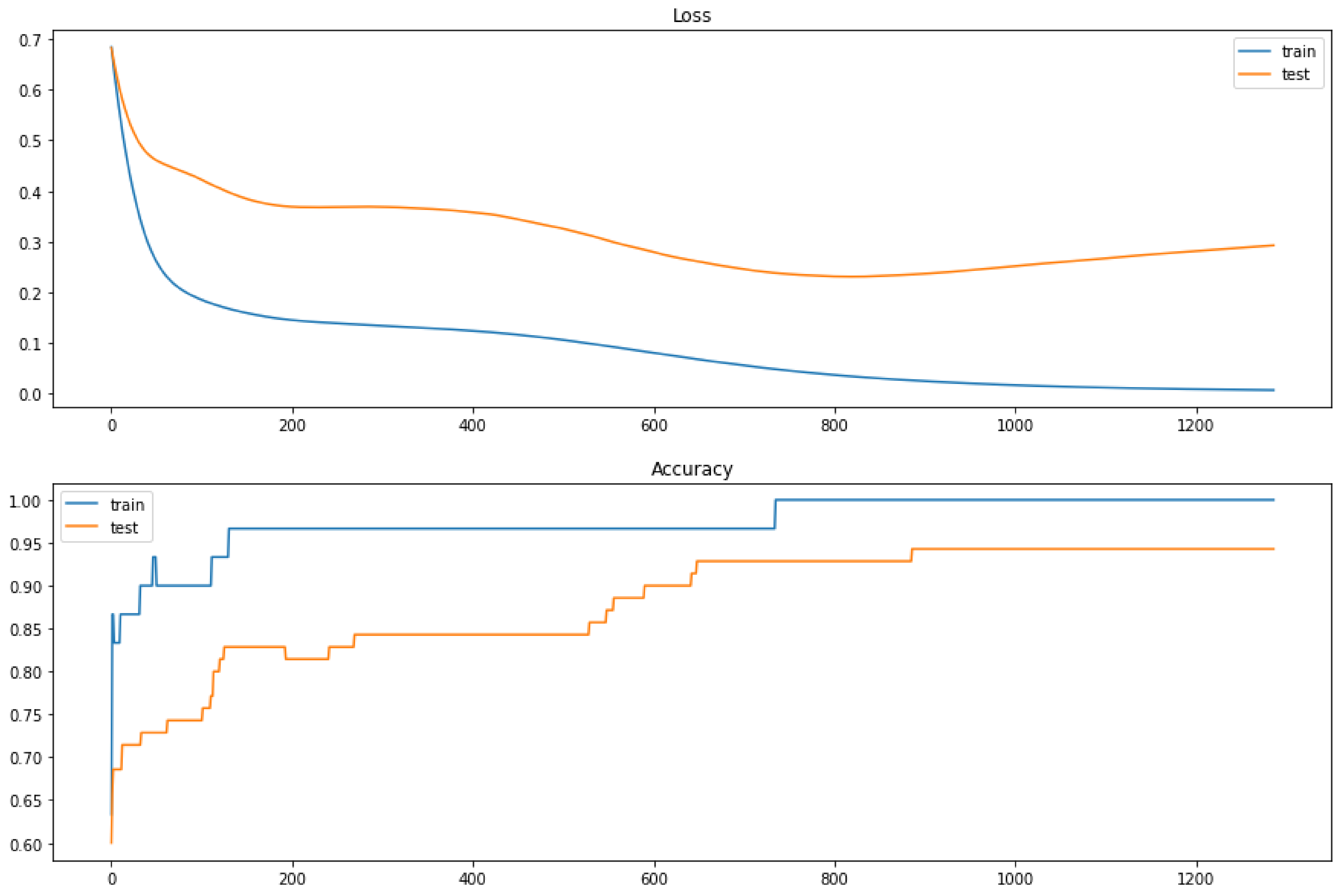
{0: 'Train: 0.83 Test: 0.76',
50: 'Train: 0.9 Test: 0.73',
100: 'Train: 0.97 Test: 0.81',
200: 'Train: 0.97 Test: 0.84',
400: 'Train: 1.0 Test: 0.94',
600: 'Train: 1.0 Test: 0.93',
800: 'Train: 1.0 Test: 0.91'}

Epoch 00403: early stopping
Train: 0.967, Test: 0.843



ES + MODEL-CHECKPOINT

Epoch 01287: early stopping
Train: 1.000, Test: 0.943



USANDO TRUE VALIDATION SET

- **Loss**

  Epoch 01275: early stopping
  Train: 0.860, Test: 0.560

- **Accuracy**

  Epoch 01035: early stopping
  Train: 1.000, Test: 0.440

REGRESSÃO

Sem Early Stopping

MSE: 33.13
STD: 28.90

Com Early Stopping

MSE: 27.87
STD: 22.61

MELHOR RESULTADO E MAIS RÁPIDO

EPOCH 00253: EARLY STOPPING
EPOCH 00215: EARLY STOPPING
EPOCH 00238: EARLY STOPPING
EPOCH 00140: EARLY STOPPING
EPOCH 00185: EARLY STOPPING
EPOCH 00289: EARLY STOPPING
EPOCH 00273: EARLY STOPPING
EPOCH 00218: EARLY STOPPING
EPOCH 00223: EARLY STOPPING
EPOCH 00286: EARLY STOPPING

Fix Vanishing Gradient

Extensões

- Inicialização dos pesos
- Algoritimo de Aprendizagem
- Visualização do Gradiente
- Incremento na complexidade de Camadas
- Incremento na complexidade de neurônios por camada

Back Propagation and Vanishing Gradients

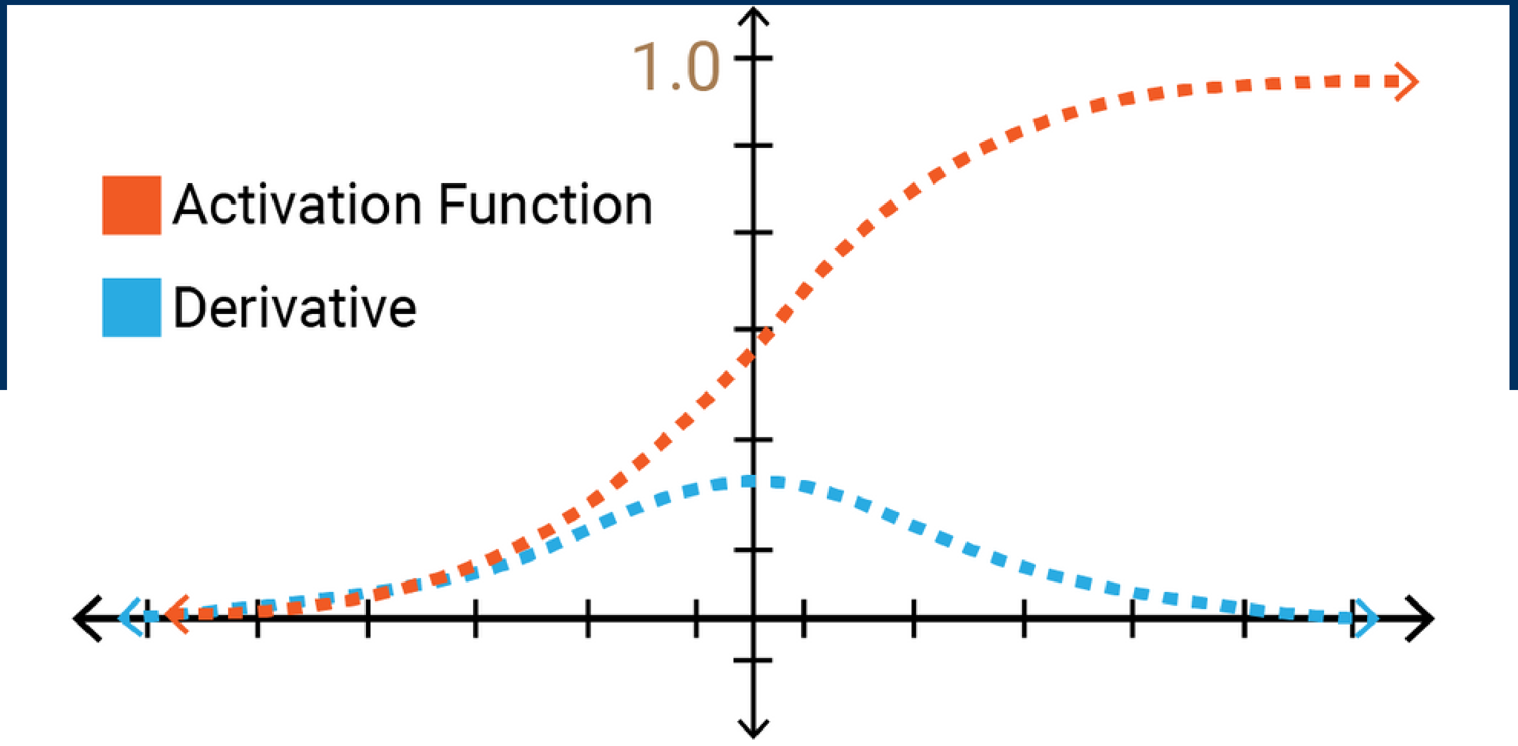
Summary: the equations of backpropagation

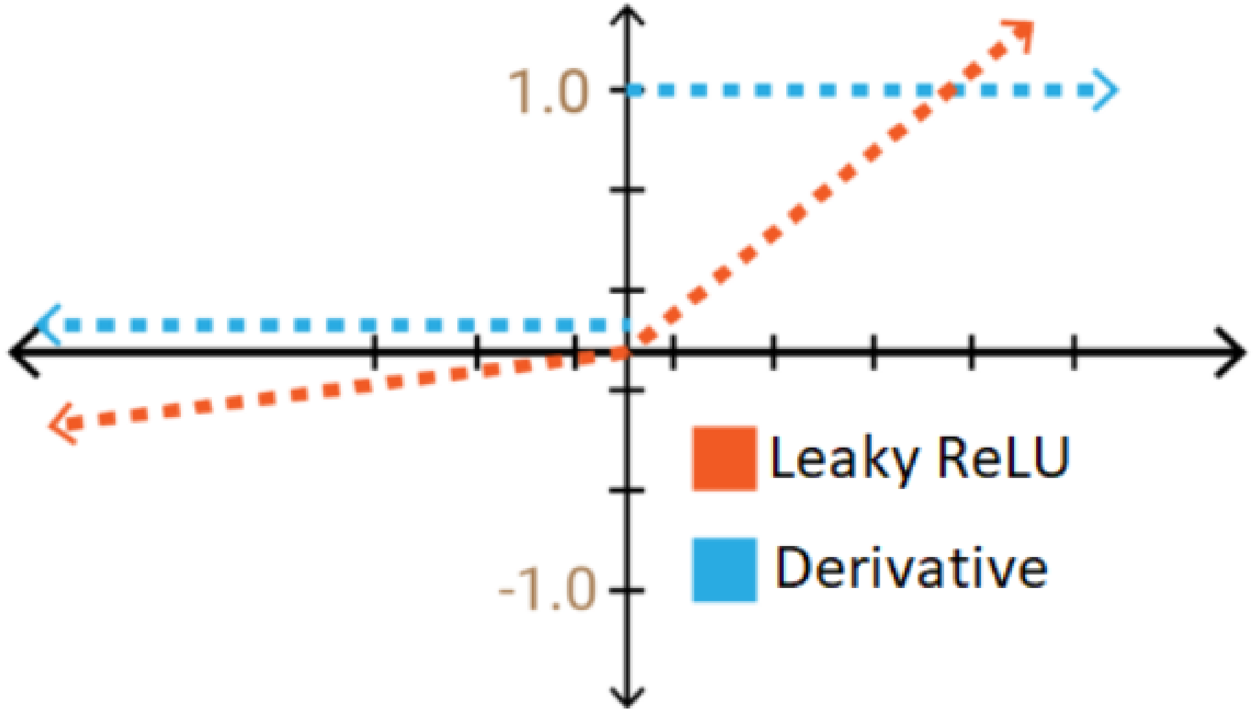
$$\delta^L = \nabla_a C \odot \sigma'(z^L) \qquad \text{(BP1)}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \qquad \text{(BP2)}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \qquad \text{(BP3)}$$

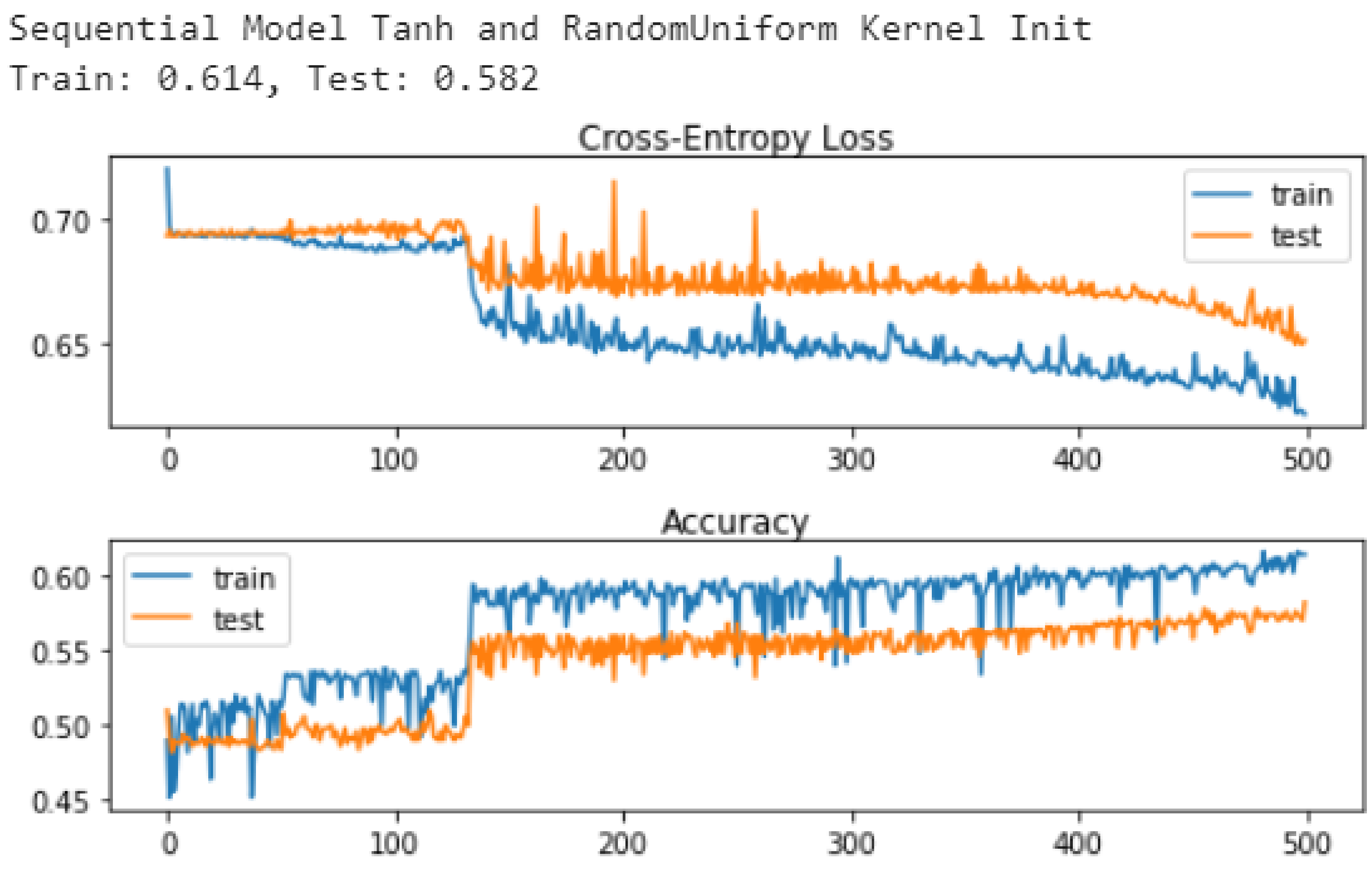
$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \qquad \text{(BP4)}$$





Deeper Neural Tanh Model

- Hidden Layers: 5
- Nodes per Layer: 5
- Activation: Tanh and Sigmoid(Out Layer)
- Kernel Initializer: Random Uniform [0, 1]
- Optimizer: SVG
    - Learning Rate: 0.01
    - Momentum: 0.9
- Loss: Binnary Cross-Entropy
- Epochs: 500



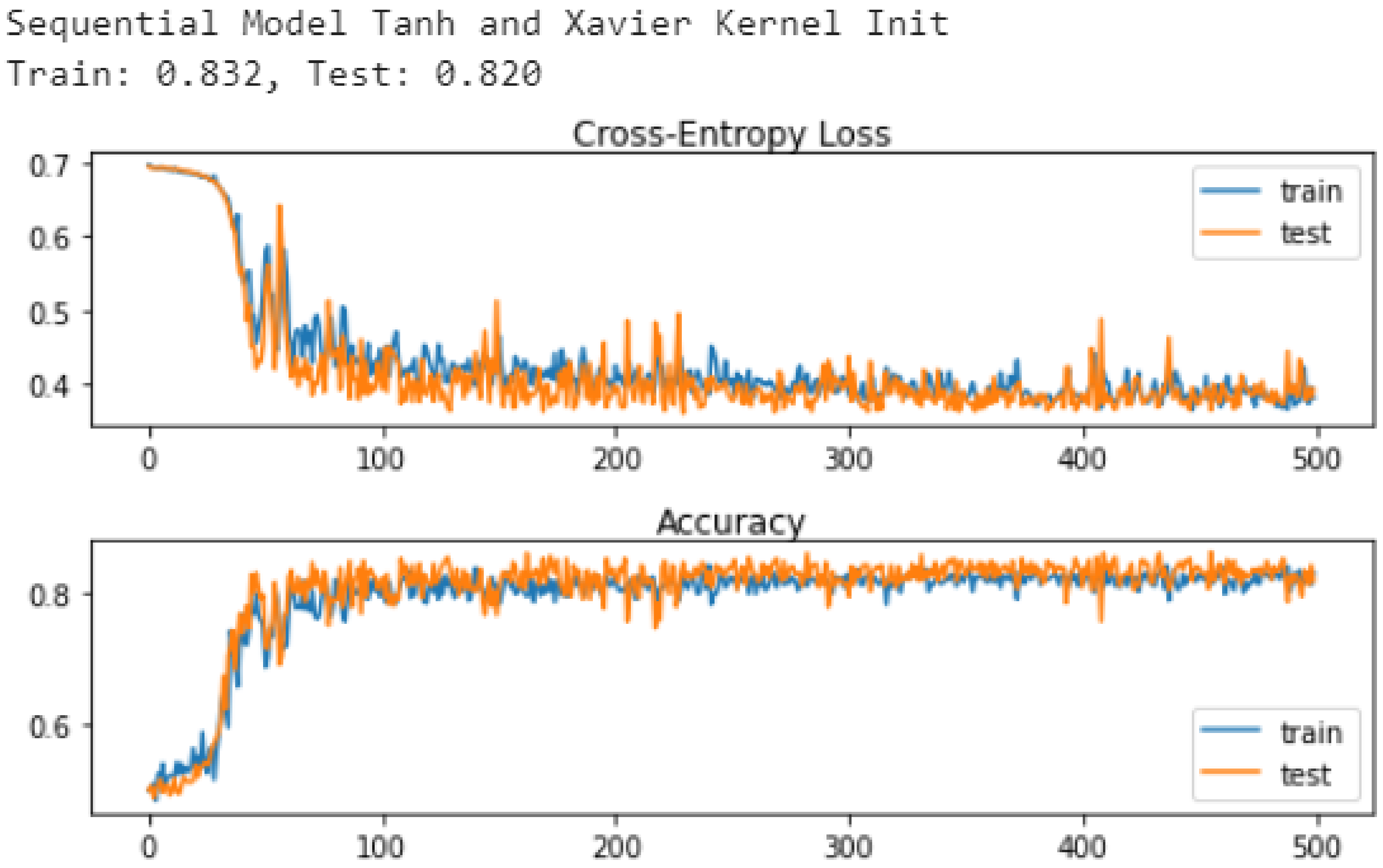
Weight Initialization

Update the Deep MLP with **tanh** activation to user **Xavier** uniform wieght initialization and report the results.

kernel_initialization=tf.keras.initializers.GlorotUniform()

$$W_{ij} \sim U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$$

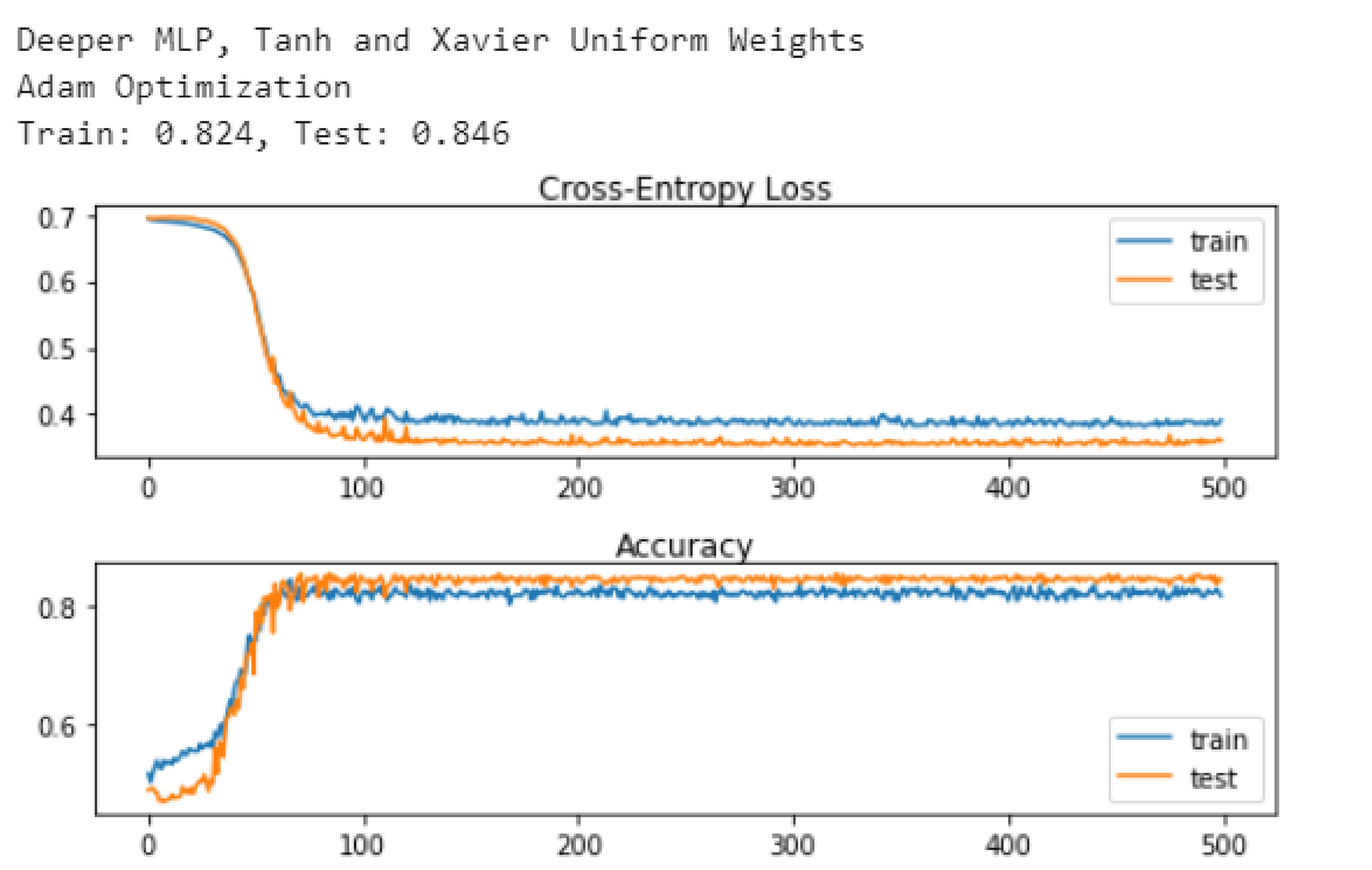
Where U is a Uniform Distribuition



Learning Algorithm

Update the deep mlp models with tanh activation to use an adaptive learning algorithm such as **Adam** and report the results.

tf.keras.optimizers.Adam(learning_rate=0.001,beta_1=0.9,beta_2=0.999,epsilon=1e-07,amsgrad=False,name="Adam",)

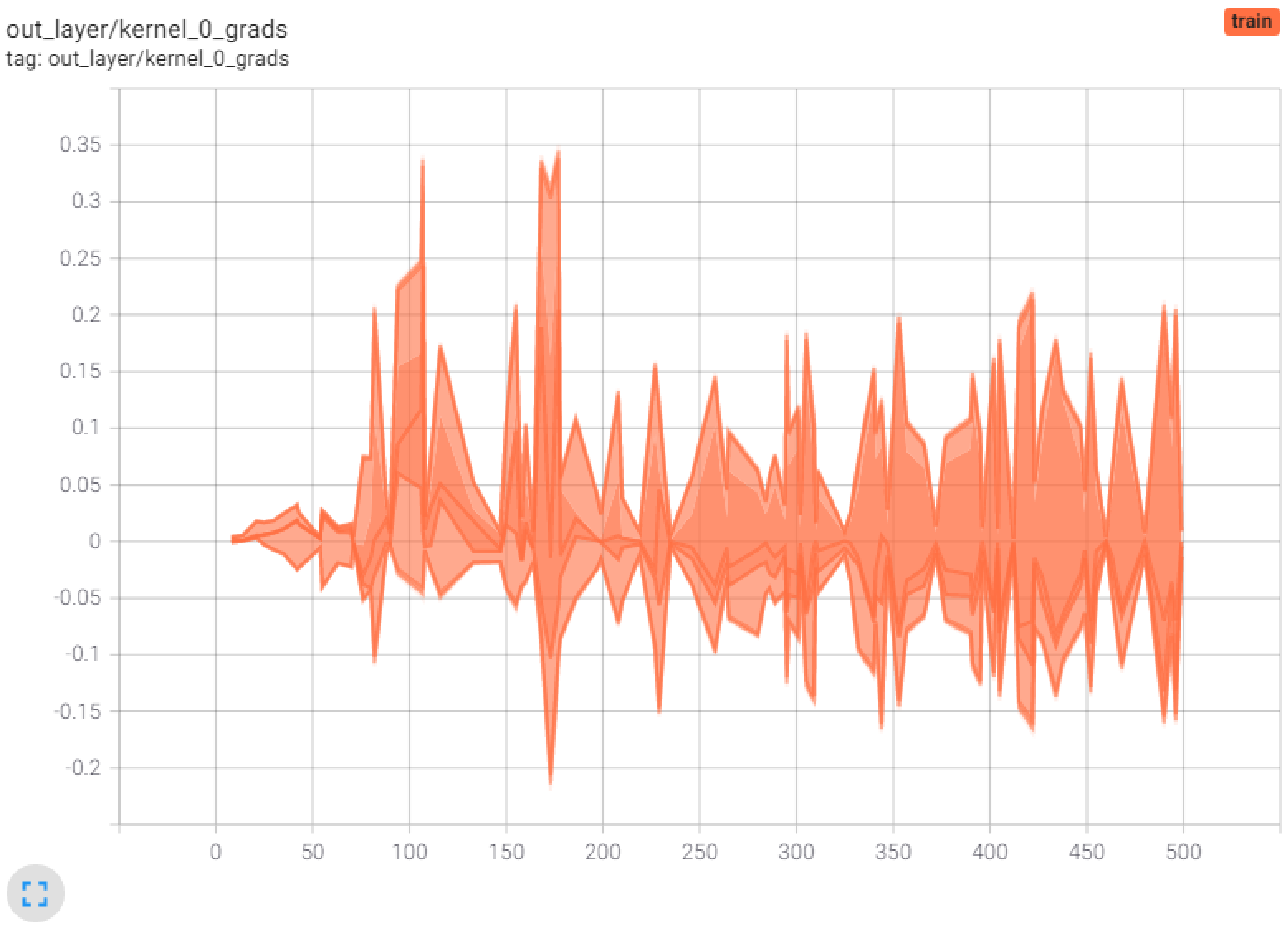


Gradient Visualization

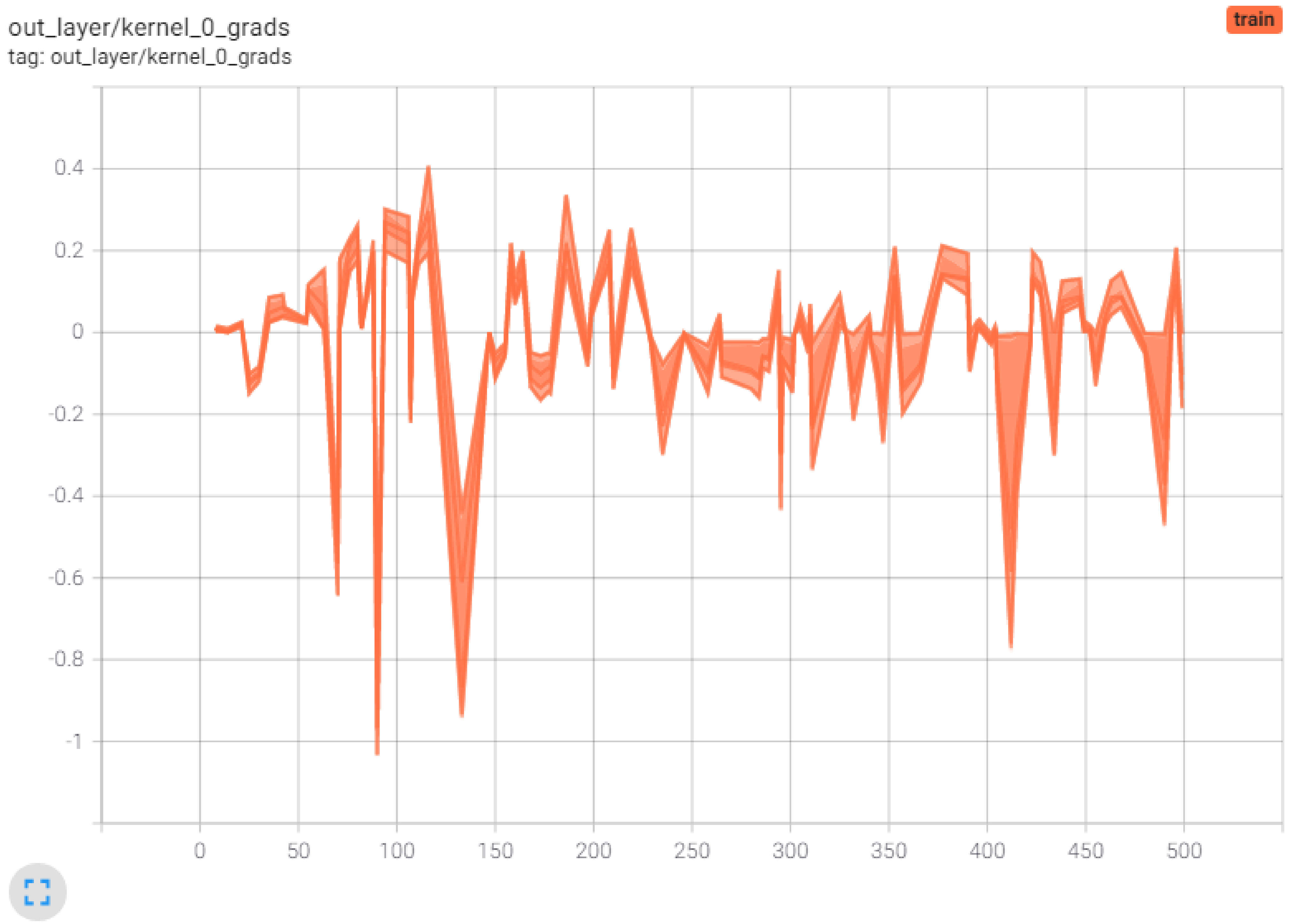
Update the **tanh** and **relu** examples to record and visualize gradients using **TensorBoard**

```
[3]
class ExtendedTensorBoard(tf.keras.callbacks.TensorBoard):
  def _log_gradients(self, epoch):
    step = tf.cast(epoch, dtype=tf.int64)
    writer = self._train_writer

    with writer.as_default(), tf.GradientTape() as g:
      _x_batch = x_train[:100]
      _y_batch = y_train[:100]

      g.watch(tf.convert_to_tensor(_x_batch))
      _y_pred = self.model(_x_batch)
      loss = self.model.loss(y_true=_y_batch, y_pred=_y_pred[0])
      gradients = g.gradient(loss, self.model.trainable_weights)


      for weights, grads in zip(self.model.trainable_weights, gradients):
        tf.summary.histogram(
            weights.name.replace(':', '_')+'_grads', data=grads, step=step)

    writer.flush()

  def on_epoch_end(self, epoch, logs=None):
    super(ExtendedTensorBoard, self).on_epoch_end(epoch, logs=logs)

    if self.histogram_freq and epoch % self.histogram_freq == 0:
      self._log_gradients(epoch)
```

Gradient Visualization Tanh



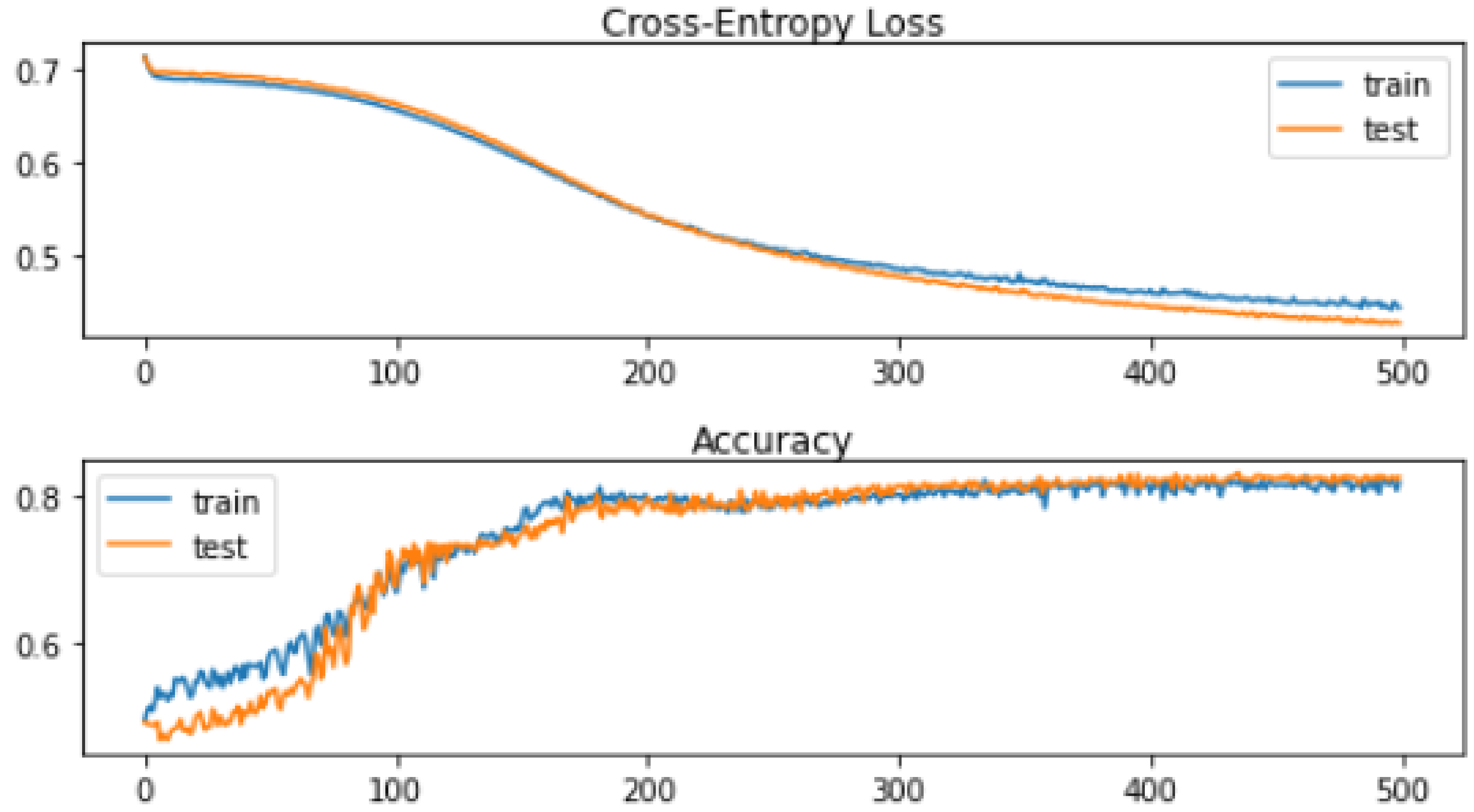
Gradient Visualization ReLU



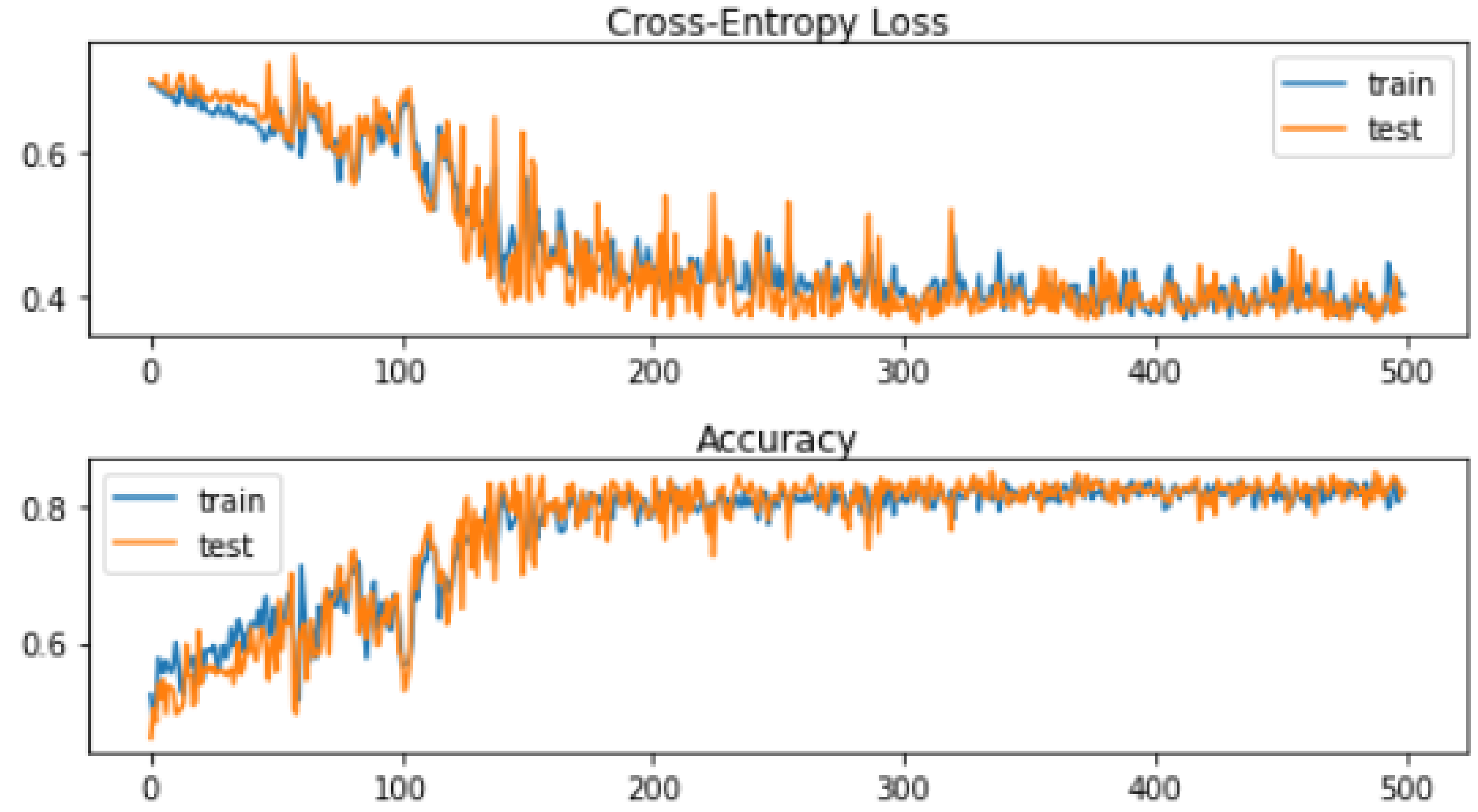
Study Model Depth

Create an experiment using the mlp with tanh activation and report the perfomance of models as the number of hidden layers is sicreased from 1 to 10.

```
Model Depth 1
Train: 0.822, Test: 0.826
```



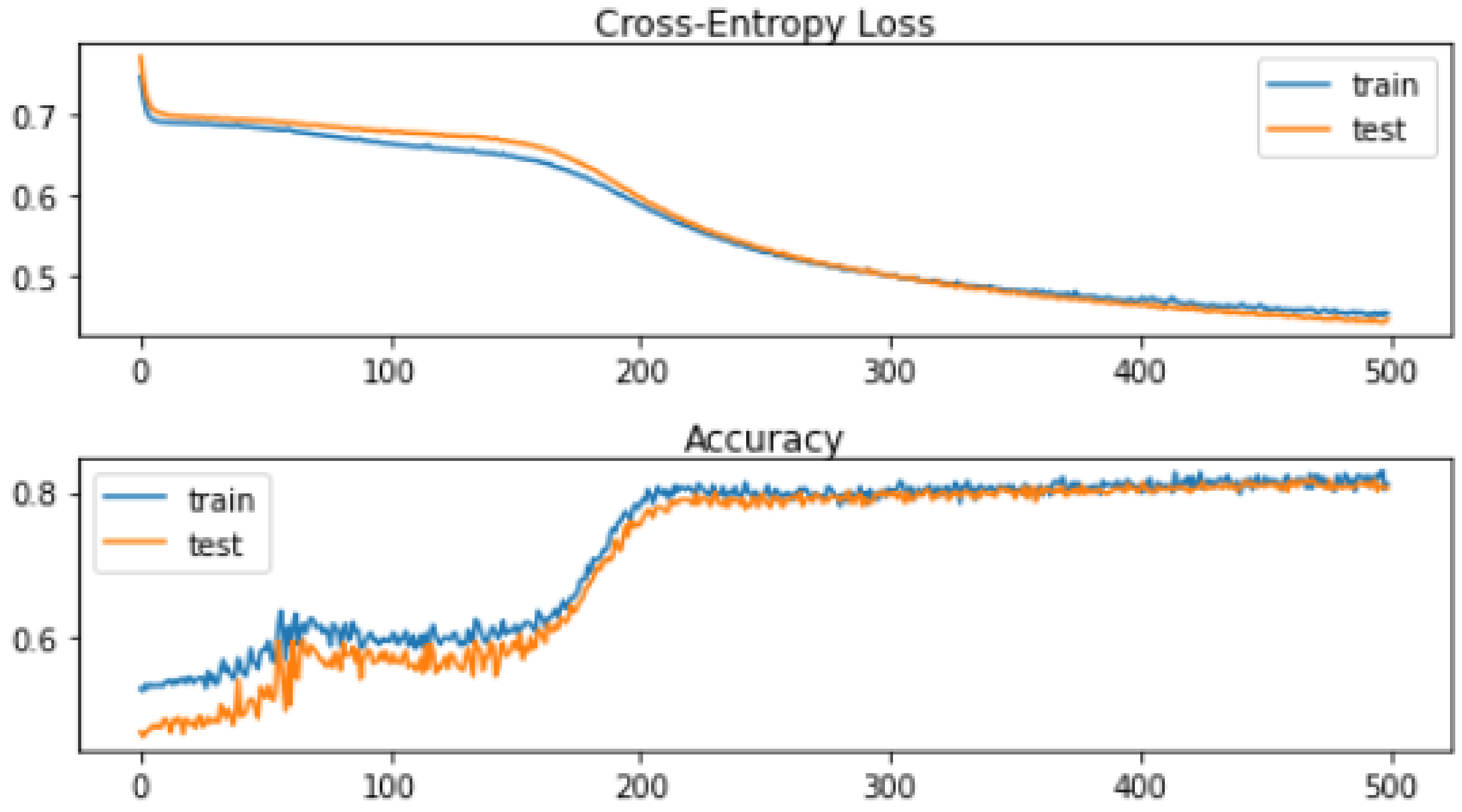

```
Model Depth 10
Train: 0.826, Test: 0.826
```



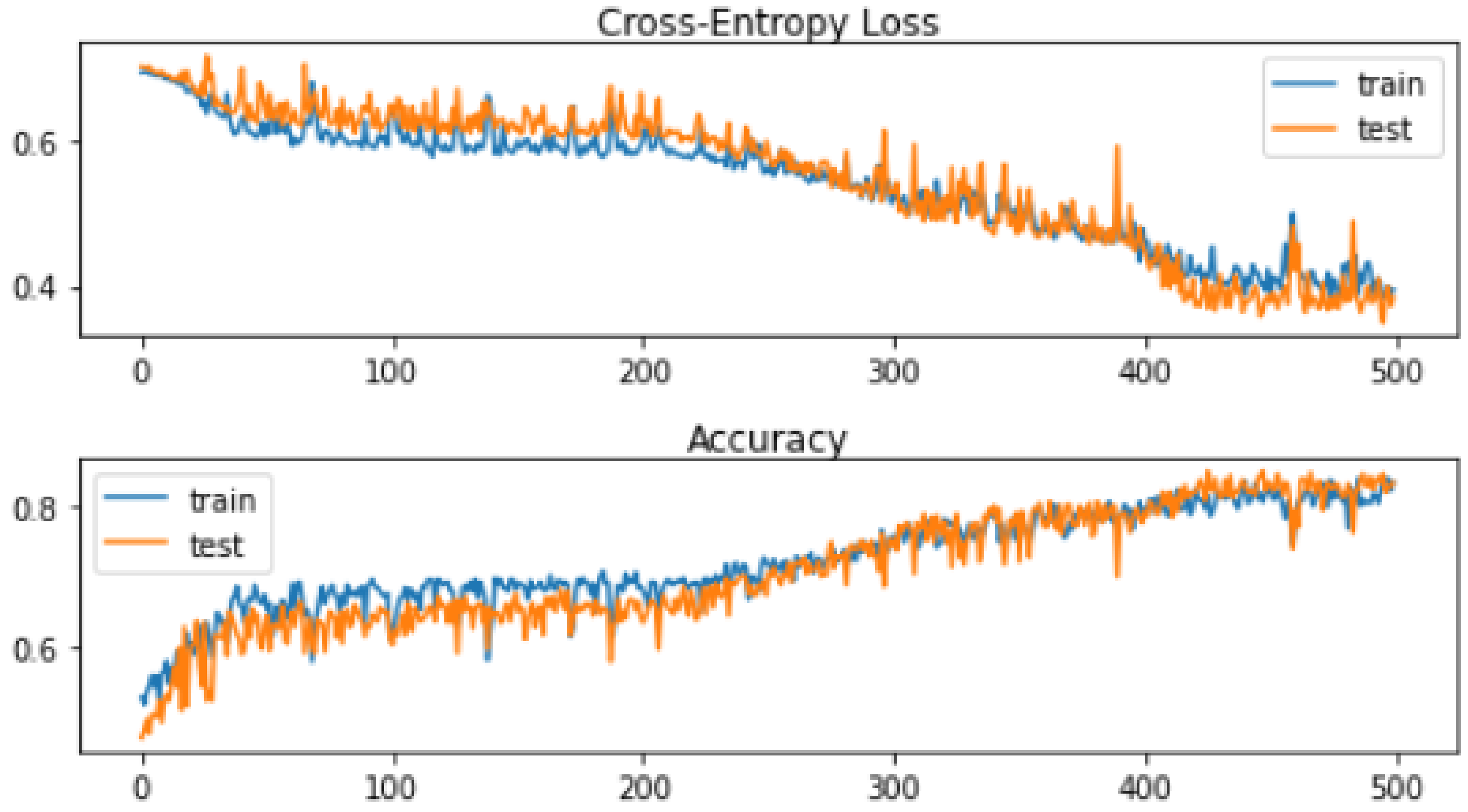
Increase Breadth

Increase the number of nodes in the hidden layers of the mlp with tanh activation from 5 to 25 and report performance as number of layers area increased from 1 to 10

```
Model Breadth 5
Model Depth 1
Train: 0.820, Test: 0.806
```



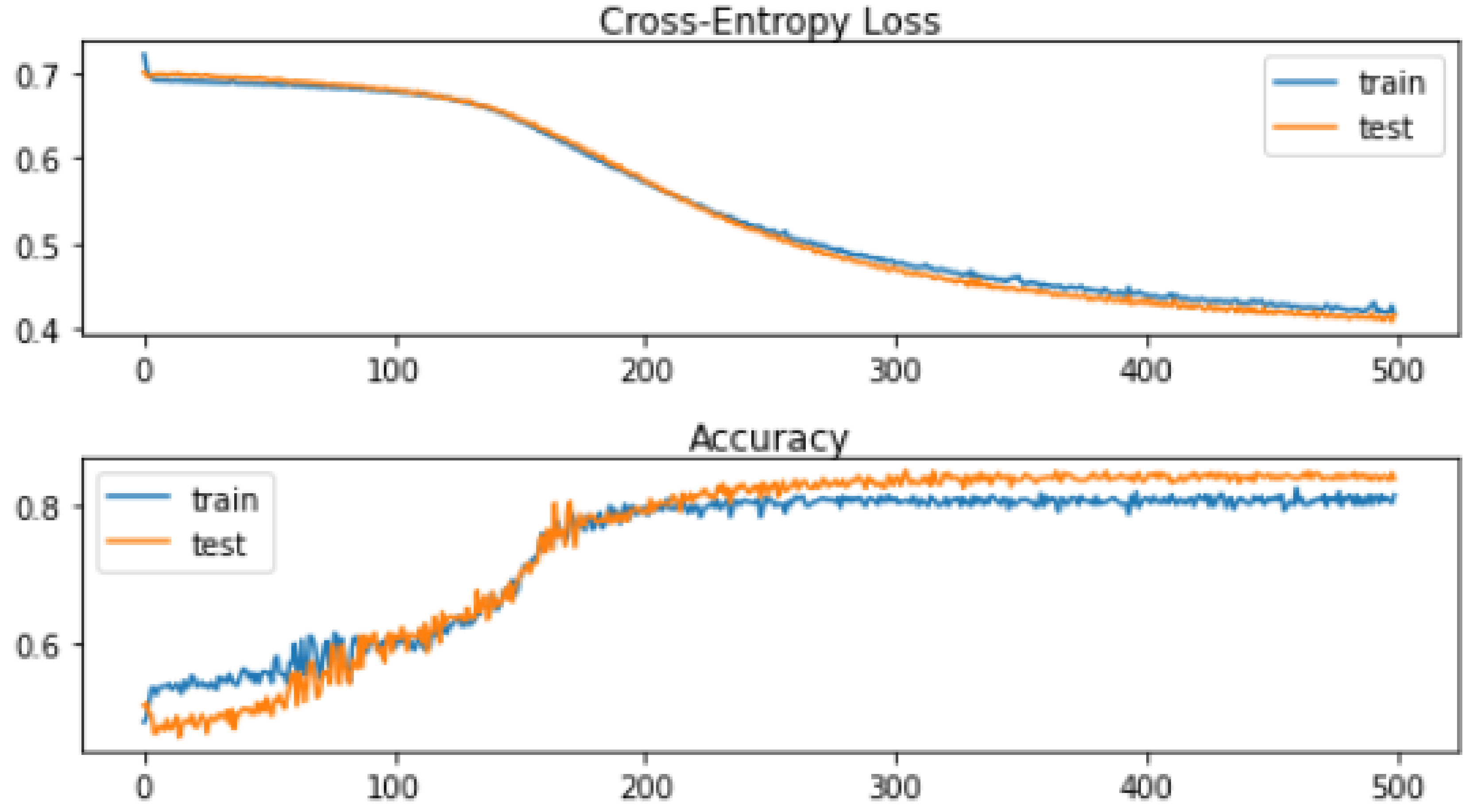

```
Model Breadth 5
Model Depth 10
Train: 0.826, Test: 0.836
```



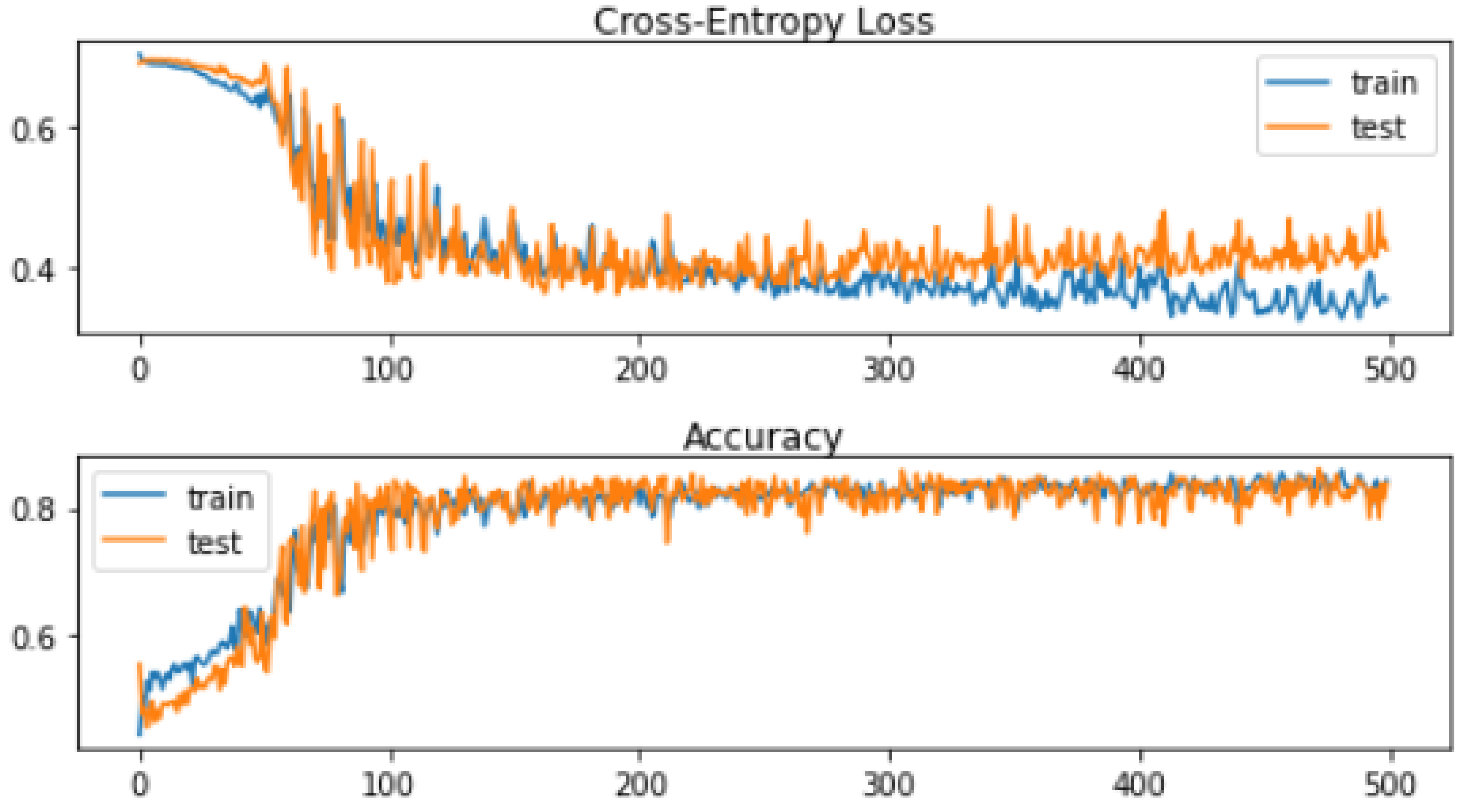
Increase Breadth

Increase the number of nodes in the hidden layers of the mlp with tanh activation from 5 to 25 and report performance as number of layers area increased from 1 to 10

```
Model Breadth 10
Model Depth 1
Train: 0.814, Test: 0.840
```



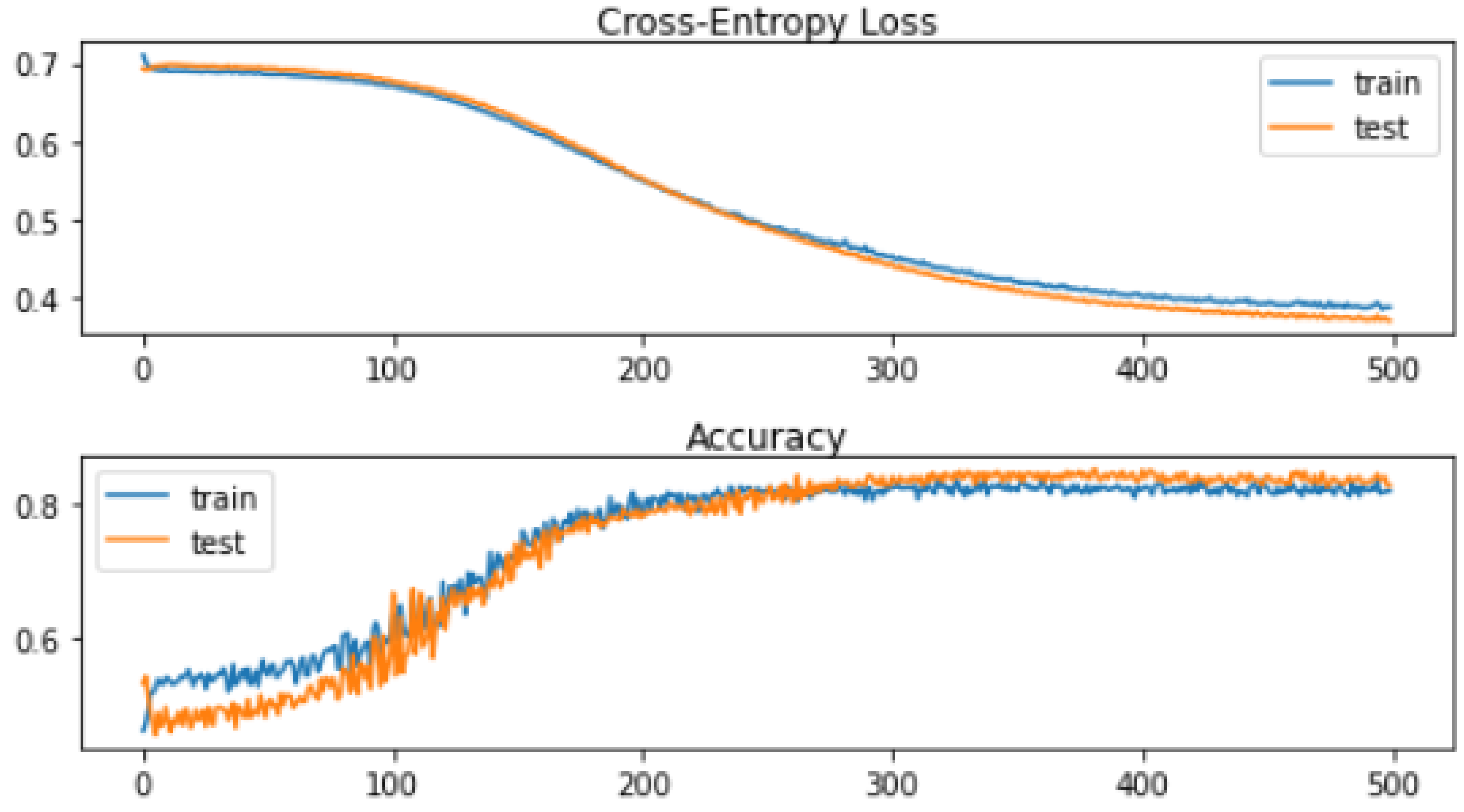

```
Model Breadth 10
Model Depth 10
Train: 0.854, Test: 0.838
```



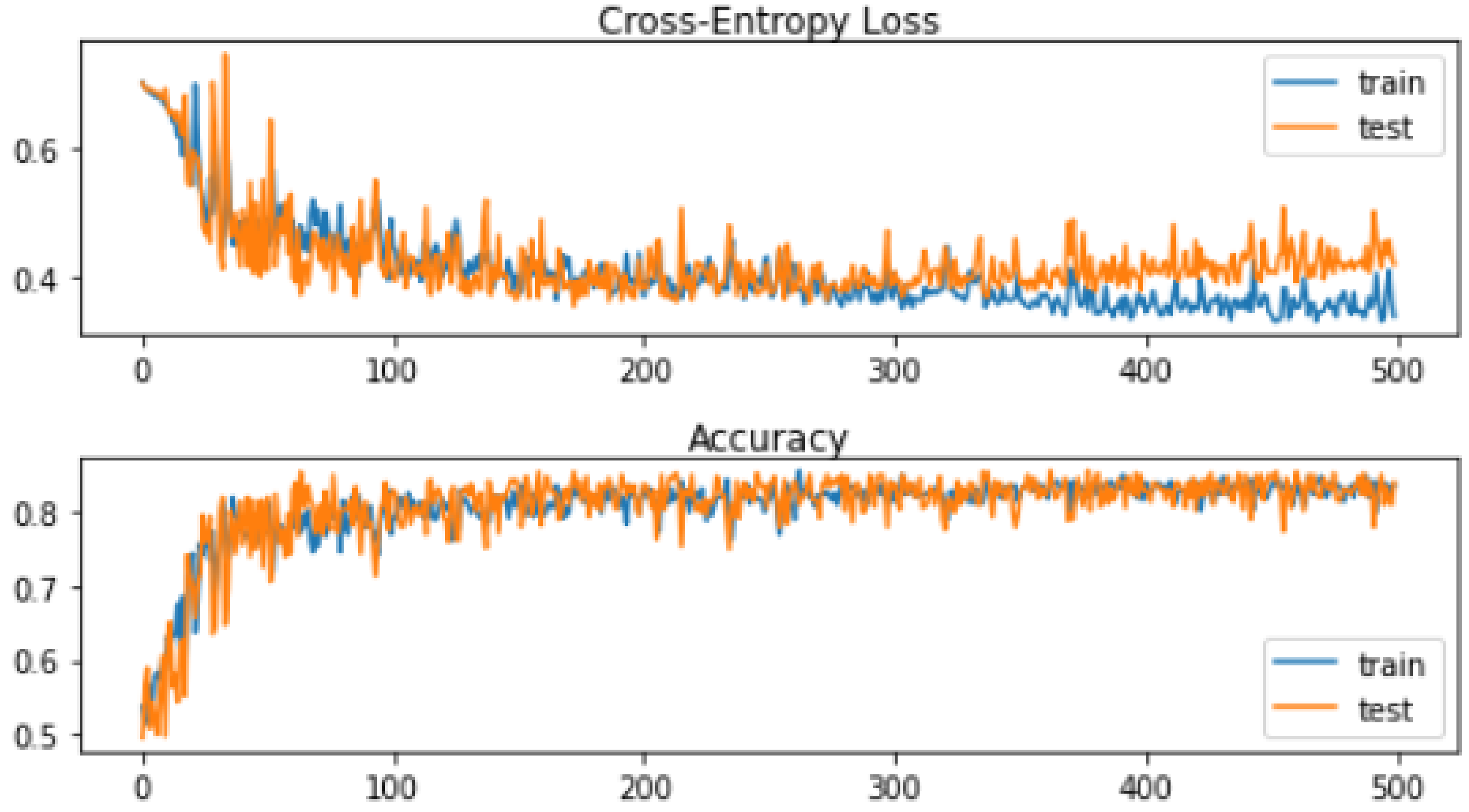
Increase Breadth

Increase the number of nodes in the hidden layers of the mlp with tanh activation from 5 to 25 and report performance as number of layers area increased from 1 to 10

```
Model Breadth 15
Model Depth 1
Train: 0.826, Test: 0.826
```



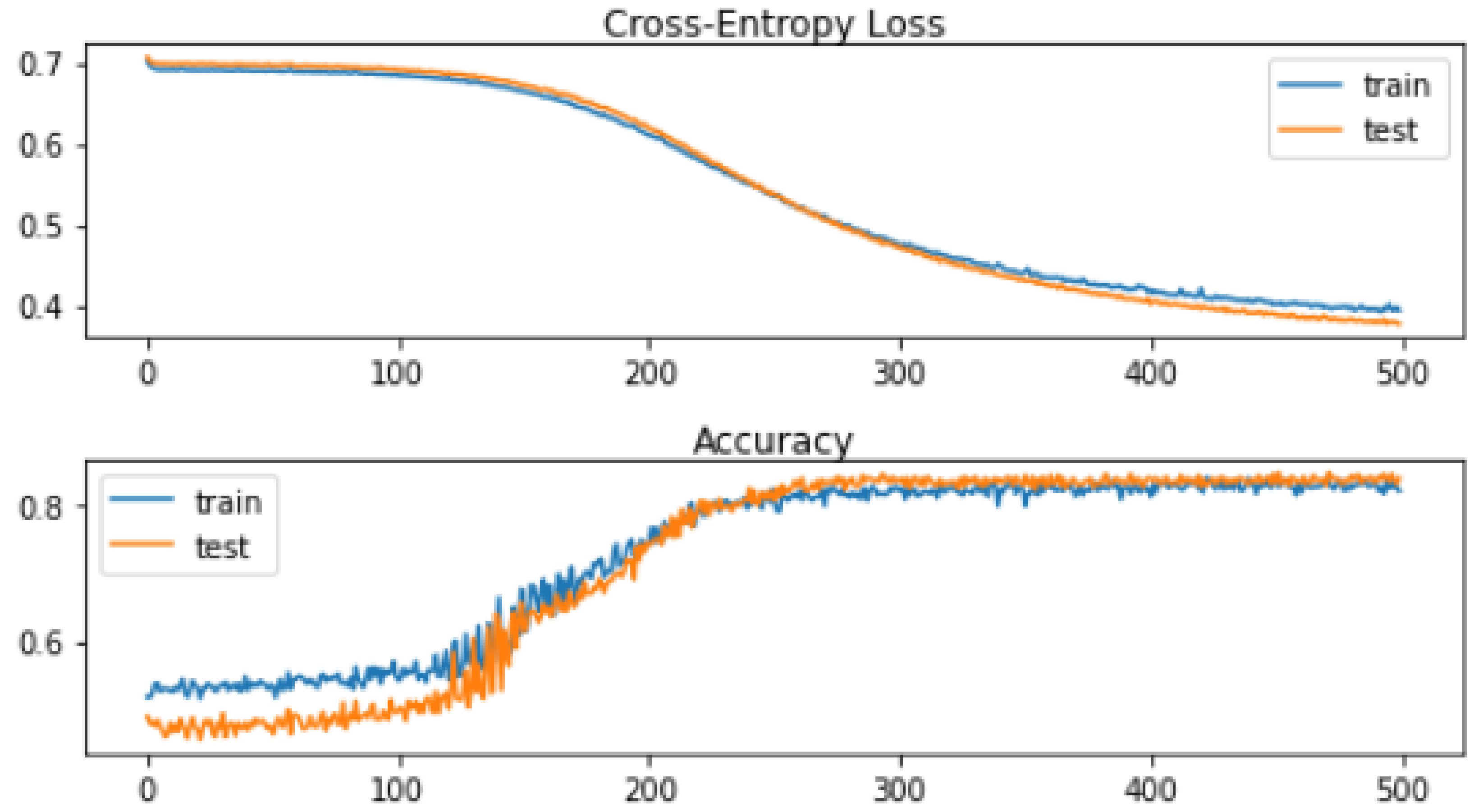

```
Model Breadth 15
Model Depth 10
Train: 0.848, Test: 0.840
```



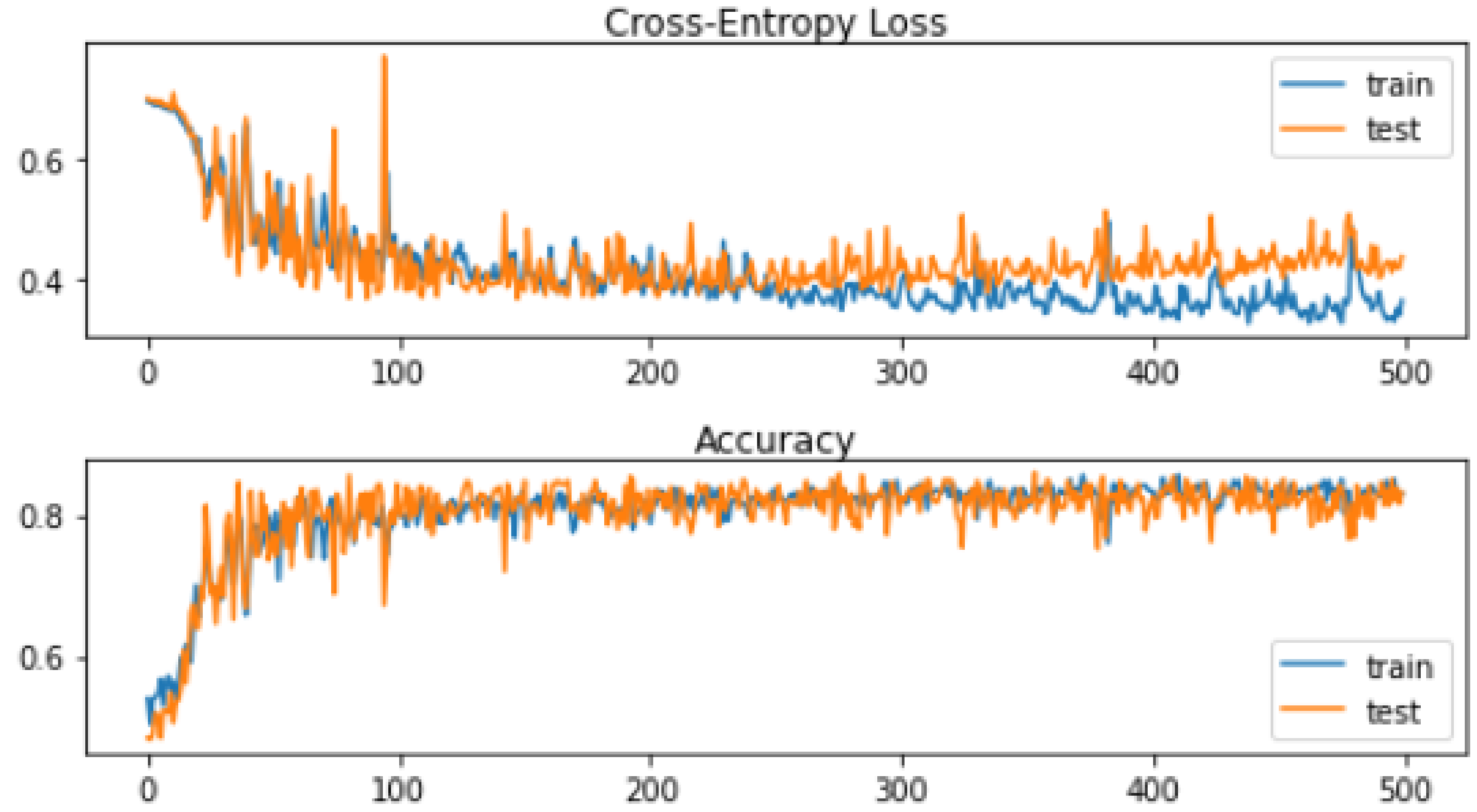
Increase Breadth

Increase the number of nodes in the hidden layers of the mlp with tanh activation from 5 to 25 and report performance as number of layers area increased from 1 to 10

```
Model Breadth 20
Model Depth 1
Train: 0.834, Test: 0.838
```



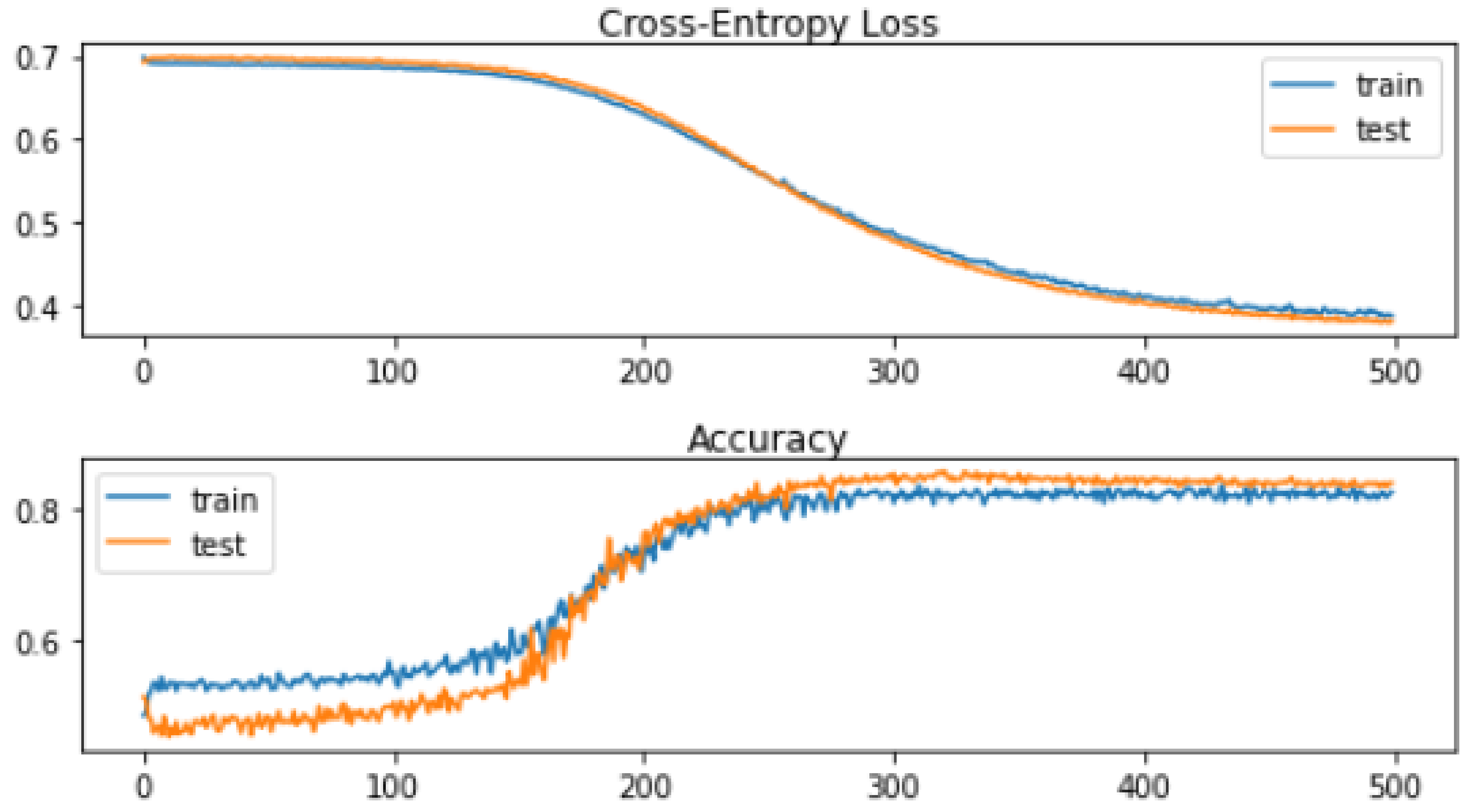

```
Model Breadth 20
Model Depth 10
Train: 0.856, Test: 0.832
```



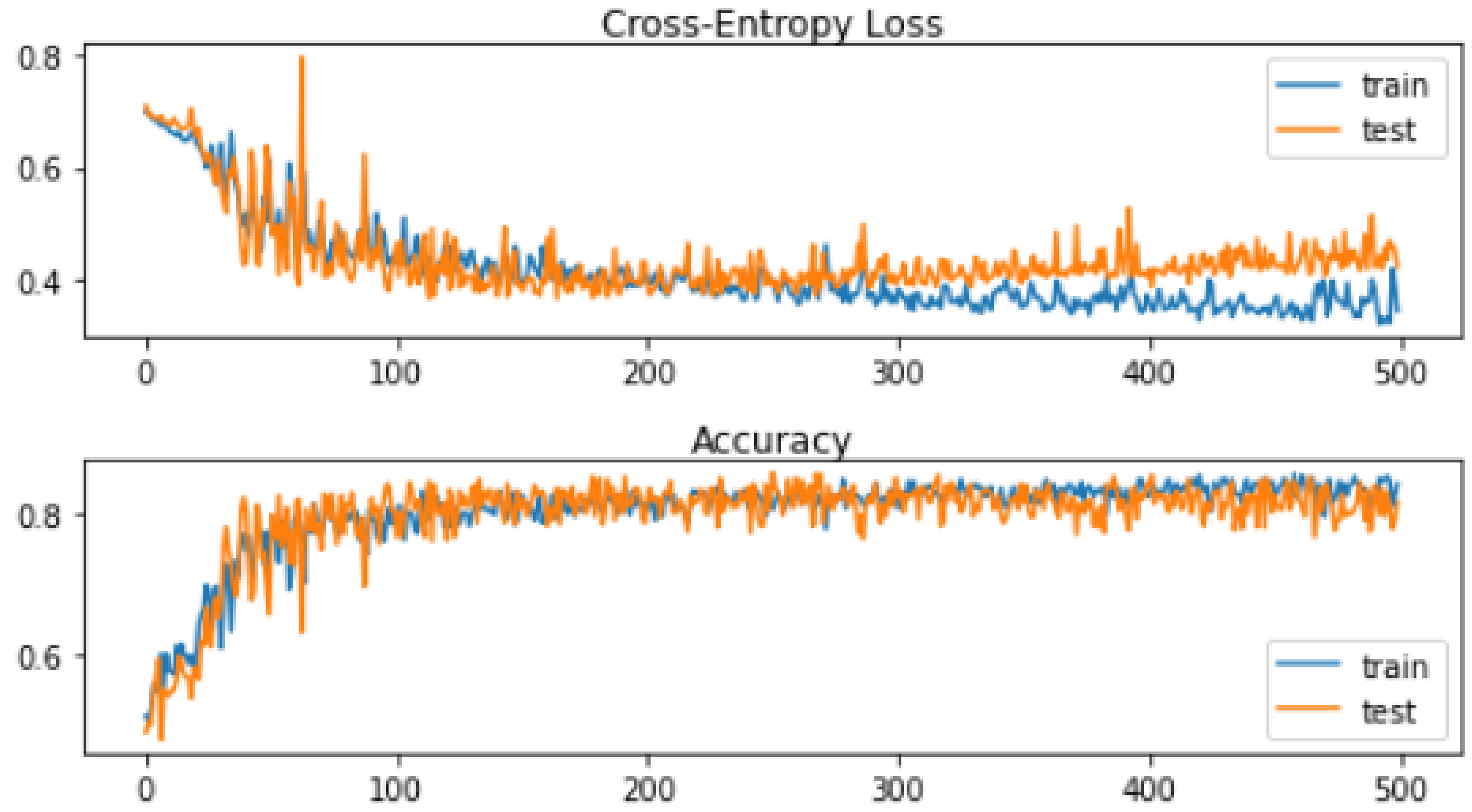
Increase Breadth

Increase the number of nodes in the hidden layers of the mlp with tanh activation from 5 to 25 and report performance as number of layers area increased from 1 to 10

```
Model Breadth 25
Model Depth 1
Train: 0.818, Test: 0.838
```




```
Model Breadth 25
Model Depth 10
Train: 0.852, Test: 0.818
```



References

1. **Xavier Kernel Initialization**
   a. https://paperswithcode.com/method/xavier-initialization
2. **Vanishing Gradients**
   a. https://towardsdatascience.com/the-problem-of-vanishing-gradients-68cea05e2625
3. **Back Propagation**
   a. http://neuralnetworksanddeeplearning.com/chap2.html
4. **Visualization of Gradients on TensorBoard**
   a. https://github.com/tensorflow/tensorflow/issues/31542