7

May 2, 2019

```
In [355]: import keras
          import pandas as pd
          import numpy as np
          from keras import Sequential
          import matplotlib.pyplot as plt
          from keras.layers import Dense, Conv1D, MaxPooling1D, Flatten, Input, LSTM, Embeddin
          import tensorflow as tf
          from keras.backend.tensorflow_backend import set_session
          config = tf.ConfigProto()
          config.gpu_options.allow_growth = True
          config.log_device_placement = True
          sess = tf.Session(config=config)
          set_session(sess)
          sess.as_default()
          sess.graph.as_default()
```

```
Out[355]: <contextlib._GeneratorContextManager at 0x7f8e095763c8>
```

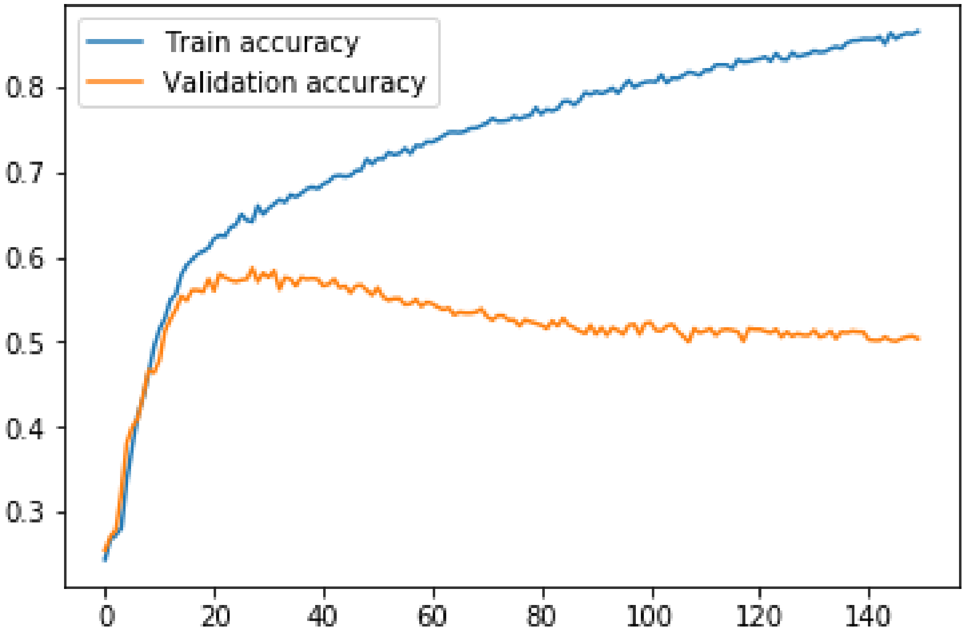
```
In [25]: data = np.genfromtxt('seqtrain.csv', delimiter=',')
         train_X, train_Y = data[:, :29], data[:, 29:]
         print(train_X.shape, train_Y.shape)
```

```
(4000, 29) (4000, 4)
```

```
In [32]: test_data = np.genfromtxt('seqvalid.csv', delimiter=',')
         test_X, test_Y = test_data[:, :29], test_data[:, 29:]
         print(test_X.shape, test_Y.shape)
```

```
(1000, 29) (1000, 4)
```

```
In [211]: model = Sequential()
          model.add(Dense(units=50, activation='relu', input_dim=29))
          model.add(Dense(units=25, activation='relu', input_dim=29))
          model.add(Dense(units=10, activation='relu', input_dim=29))
          model.add(Dense(units=5, activation='relu', input_dim=29))
          model.add(Dense(units=4, activation='softmax'))

          model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'
In [130]: history = model.fit(train_X, train_Y, epochs=150, batch_size=128, validation_data=(te

In [131]: model.evaluate(test_X, test_Y)

1000/1000 [==============================] - 0s 63us/step


Out[131]: [1.8841854248046874, 0.504]

In [132]: plt.plot(history.history['acc'], label='Train accuracy')
          plt.plot(history.history['val_acc'], label='Validation accuracy')
          plt.legend()

Out[132]: <matplotlib.legend.Legend at 0x7f8e61fc2668>
```

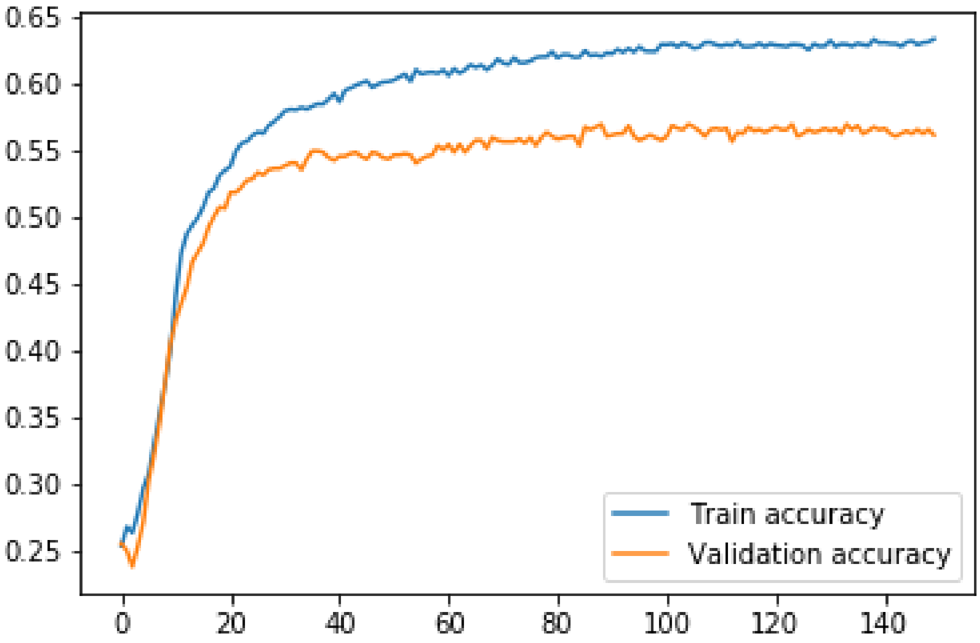


```
In [445]: model = Sequential()
          model.add(Dense(units=14, activation='relu', input_dim=29))
          model.add(Dense(units=7, activation='relu', input_dim=29))
          model.add(Dense(units=4, activation='softmax'))

          model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']
          history = model.fit(train_X, train_Y, epochs=150, batch_size=128, validation_data=(te

In [446]: model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_292 (Dense)            (None, 14)                420
_________________________________________________________________
dense_293 (Dense)            (None, 7)                 105
_________________________________________________________________
dense_294 (Dense)            (None, 4)                 32
=================================================================
Total params: 557
Trainable params: 557
Non-trainable params: 0
_________________________________________________________________
```

```
In [435]: plt.plot(history.history['acc'], label='Train accuracy')
          plt.plot(history.history['val_acc'], label='Validation accuracy')
          plt.legend()
```

```
Out[435]: <matplotlib.legend.Legend at 0x7f8dcf2d7eb8>
```



```
In [313]: model.summary()
```

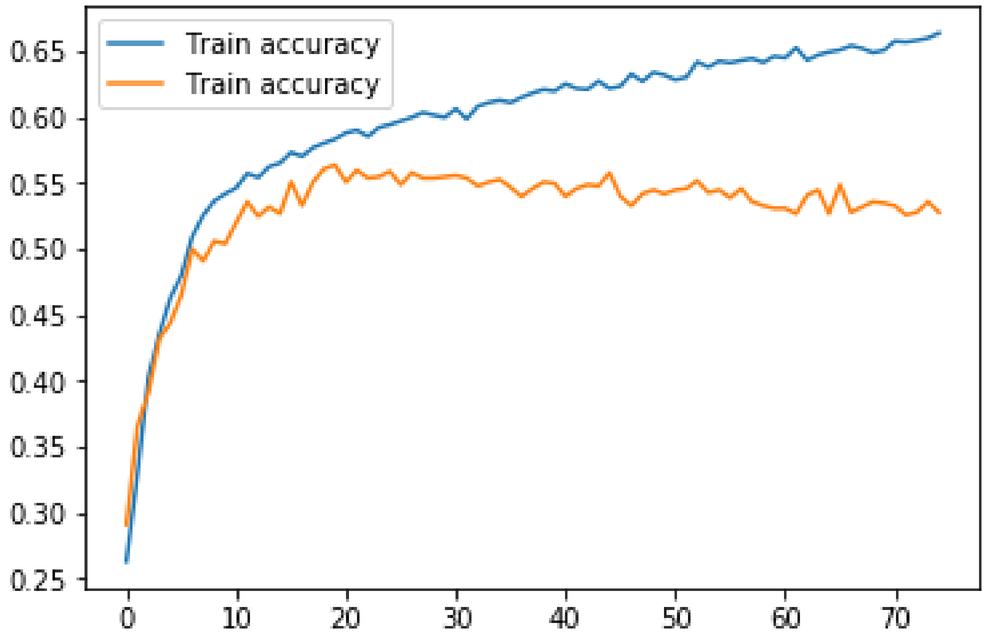
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_72 (Conv1D)           (None, 29, 3)             12
_________________________________________________________________
conv1d_73 (Conv1D)           (None, 29, 7)             70
_________________________________________________________________
conv1d_74 (Conv1D)           (None, 29, 14)            308
_________________________________________________________________
conv1d_75 (Conv1D)           (None, 29, 28)            420
_________________________________________________________________
flatten_23 (Flatten)         (None, 812)               0
_________________________________________________________________
dense_196 (Dense)            (None, 10)                8130
_________________________________________________________________
dense_197 (Dense)            (None, 5)                 55
_________________________________________________________________
dense_198 (Dense)            (None, 4)                 24
=================================================================
Total params: 9,019
Trainable params: 9,019
Non-trainable params: 0
_________________________________________________________________
```

```
In [336]: model = Sequential()

          model.add(Conv1D(filters=3, kernel_size=3, activation='relu', padding='same', input_s

          model.add(Conv1D(filters=7, kernel_size=5, activation='relu', padding='same'))
          model.add(Conv1D(filters=14, kernel_size=3, activation='relu', padding='same'))
          model.add(Flatten())
          model.add(Dense(10, activation='relu'))
          model.add(Dense(5, activation='relu'))
          model.add(Dense(4, activation='softmax'))

          model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']
```

```
In [337]: history = model.fit(np.expand_dims(train_X, axis=2), train_Y, epochs=75, batch_size=6
```

```
In [338]: plt.plot(history.history['acc'], label='Train accuracy')
          plt.plot(history.history['val_acc'], label='Train accuracy')
          plt.legend()
```

```
Out[338]: <matplotlib.legend.Legend at 0x7f8e0967f898>
```



```
In [339]: model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_96 (Conv1D)           (None, 29, 3)             12
_________________________________________________________________
conv1d_97 (Conv1D)           (None, 29, 7)             112
_________________________________________________________________
conv1d_98 (Conv1D)           (None, 29, 14)            308
_________________________________________________________________
flatten_33 (Flatten)         (None, 406)               0
_________________________________________________________________
dense_218 (Dense)            (None, 10)                4070
_________________________________________________________________
dense_219 (Dense)            (None, 5)                 55
_________________________________________________________________
dense_220 (Dense)            (None, 4)                 24
=================================================================
Total params: 4,581
Trainable params: 4,581
Non-trainable params: 0
_________________________________________________________________
```

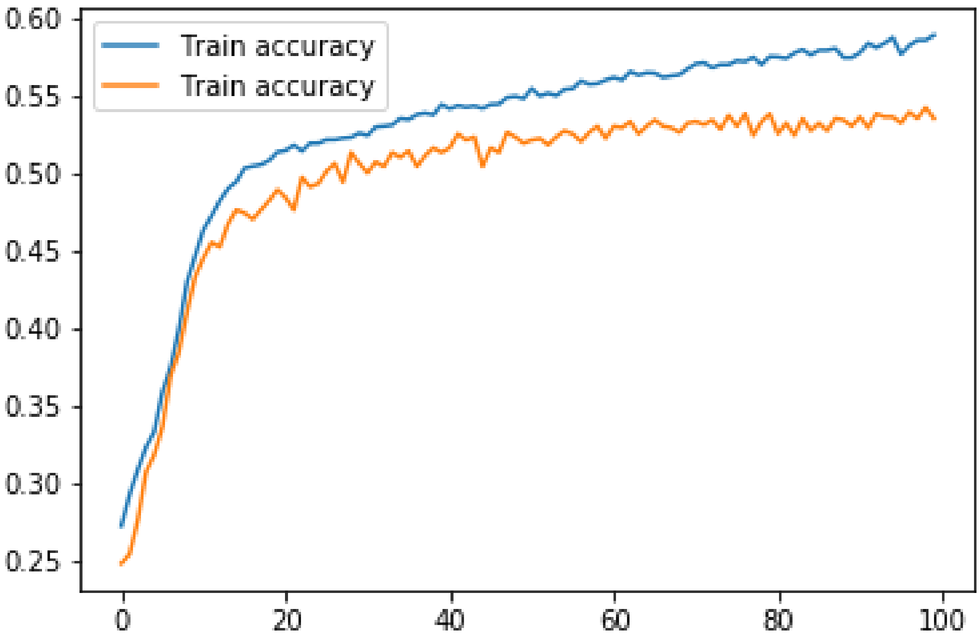
```
In [340]: model = Sequential()
```

```
          model.add(Conv1D(filters=3, kernel_size=3, activation='relu', padding='same', input_
          model.add(Conv1D(filters=3, kernel_size=3, activation='relu', padding='same', input_
          model.add(Flatten())
          model.add(Dense(5, activation='relu'))
          model.add(Dense(4, activation='softmax'))

          model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'
```

```
In [334]: history = model.fit(np.expand_dims(train_X, axis=2), train_Y, epochs=50, batch_size=
```

```
In [335]: plt.plot(history.history['acc'], label='Train accuracy')
          plt.plot(history.history['val_acc'], label='Train accuracy')
          plt.legend()
```

```
Out[335]: <matplotlib.legend.Legend at 0x7f8e09e1d518>
```



```
In [385]: model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_13 (Embedding)     (None, None, 290)         8410
_________________________________________________________________
lstm_41 (LSTM)               (None, None, 128)         214528
_________________________________________________________________
lstm_42 (LSTM)               (None, None, 64)          49408
```

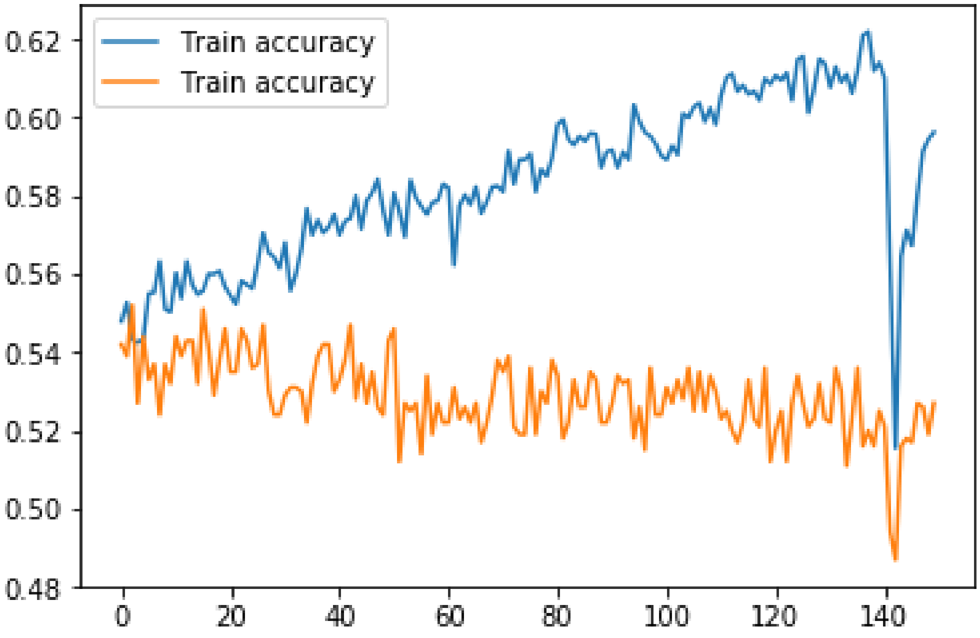
```
_________________________________________________________________
dense_242 (Dense)            (None, None, 15)          975       
_________________________________________________________________
dense_243 (Dense)            (None, None, 4)           64        
=================================================================
Total params: 273,385
Trainable params: 273,385
Non-trainable params: 0
_________________________________________________________________
```

```
In [422]: model = Sequential()

          model.add(Embedding(29, output_dim=29))
          model.add(LSTM(29, activation='relu'))
          model.add(Dense(10, activation='relu'))
          model.add(Dense(4, activation='softmax'))

          model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']
```

```
In [424]: history = model.fit(train_X, train_Y, epochs=150, batch_size=128, verbose=0, validat
```

```
In [425]: plt.plot(history.history['acc'], label='Train accuracy')
          plt.plot(history.history['val_acc'], label='Train accuracy')
          plt.legend()
```

```
Out[425]: <matplotlib.legend.Legend at 0x7f8dd04b99e8>
```



```
In [426]: model.summary()
```

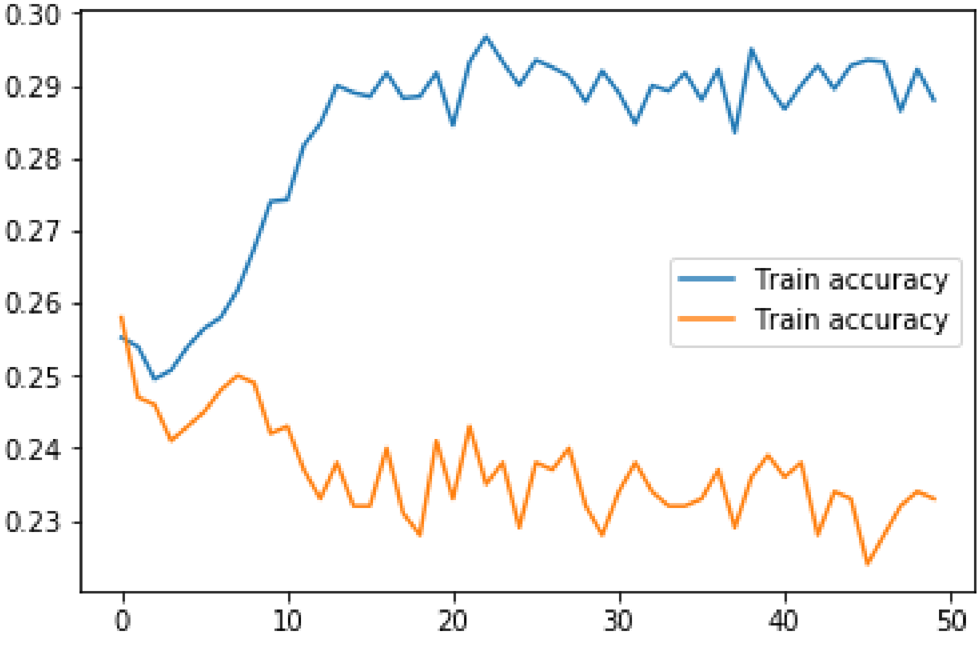
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_23 (Embedding)     (None, None, 29)          841
_________________________________________________________________
lstm_52 (LSTM)               (None, 29)                6844
_________________________________________________________________
dense_266 (Dense)            (None, 10)                300
_________________________________________________________________
dense_267 (Dense)            (None, 4)                 44
=================================================================
Total params: 8,029
Trainable params: 8,029
Non-trainable params: 0
_________________________________________________________________
```

```
In [415]: model = Sequential()
          model.add(Dense(4, activation='softmax'))
```

```
In [416]: model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']
          history = model.fit(train_X, train_Y, epochs=50, batch_size=64, verbose=0, validatio
```

```
In [417]: plt.plot(history.history['acc'], label='Train accuracy')
          plt.plot(history.history['val_acc'], label='Train accuracy')
          plt.legend()
```

```
Out[417]: <matplotlib.legend.Legend at 0x7f8dd2179780>
```



```
In [418]: model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_263 (Dense)            (None, 4)                 120
=================================================================
Total params: 120
Trainable params: 120
Non-trainable params: 0
_________________________________________________________________
```

Fully connected, CNN and RNN all reached pretty much the same about 55% accuracy on the validation set. Adding more layers or neurons really only helps to overfit and the validation accuracy never seems to go over 60% with the few architectures that I tested. Clearly the logistic regression model is really bad and at 25% accuracy it is no better than guessing, so at least a few layers are needed to achieve decent results on this dataset. Generally on this problem a smaller network seems to work best, maybe that is due to the relatively low dimensionality of the problem.

```
In [430]: model = Sequential()
          model.add(Dense(units=500, activation='relu'))
          model.add(Dense(units=250, activation='relu'))
          model.add(Dense(units=100, activation='relu'))
          model.add(Dense(units=50, activation='relu'))
          model.add(Dense(units=4, activation='softmax'))

          model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']
```

```
In [432]: history = model.fit(train_X, train_Y, epochs=50, batch_size=64, verbose=1, validatio
```

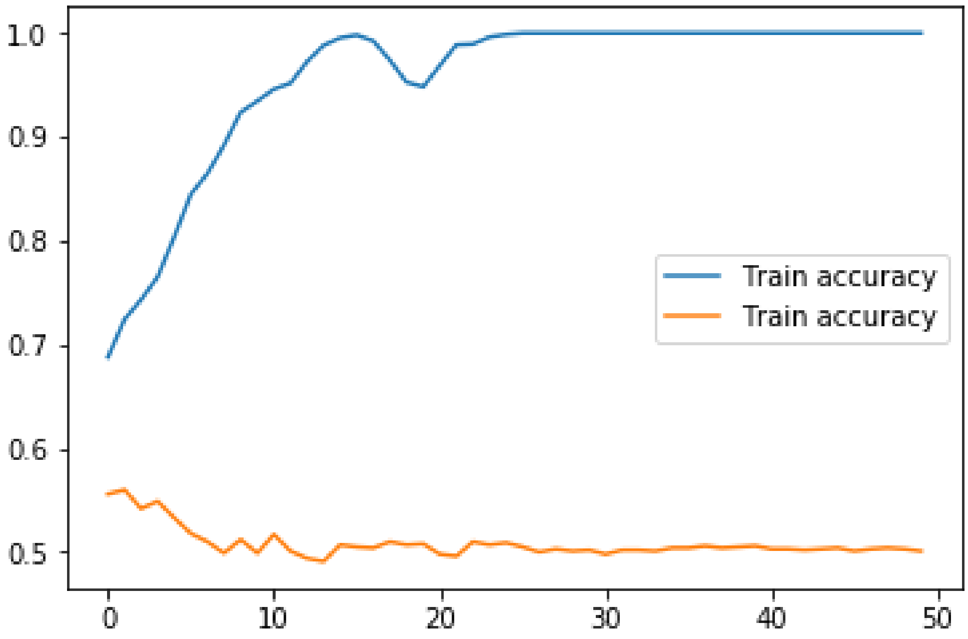
```
Train on 4000 samples, validate on 1000 samples
Epoch 1/50
4000/4000 [==============================] - 0s 89us/step - loss: 0.7201 - acc: 0.6877 - val_lo
Epoch 2/50
4000/4000 [==============================] - 0s 90us/step - loss: 0.6490 - acc: 0.7242 - val_lo
Epoch 3/50
4000/4000 [==============================] - 0s 88us/step - loss: 0.6126 - acc: 0.7433 - val_lo
Epoch 4/50
4000/4000 [==============================] - 0s 90us/step - loss: 0.5657 - acc: 0.7652 - val_lo
Epoch 5/50
4000/4000 [==============================] - 0s 89us/step - loss: 0.4823 - acc: 0.8037 - val_lo
Epoch 6/50
4000/4000 [==============================] - 0s 91us/step - loss: 0.4137 - acc: 0.8448 - val_lo
Epoch 7/50
4000/4000 [==============================] - 0s 88us/step - loss: 0.3443 - acc: 0.8648 - val_lo
```

```
Epoch 8/50
4000/4000 [==============================] - 0s 90us/step - loss: 0.2825 - acc: 0.8920 - val_lo
Epoch 9/50
4000/4000 [==============================] - 0s 90us/step - loss: 0.2262 - acc: 0.9237 - val_lo
Epoch 10/50
4000/4000 [==============================] - 0s 91us/step - loss: 0.1947 - acc: 0.9345 - val_lo
Epoch 11/50
4000/4000 [==============================] - 0s 86us/step - loss: 0.1567 - acc: 0.9460 - val_lo
Epoch 12/50
4000/4000 [==============================] - 0s 84us/step - loss: 0.1440 - acc: 0.9517 - val_lo
Epoch 13/50
4000/4000 [==============================] - 0s 88us/step - loss: 0.0948 - acc: 0.9725 - val_lo
Epoch 14/50
4000/4000 [==============================] - 0s 93us/step - loss: 0.0547 - acc: 0.9882 - val_lo
Epoch 15/50
4000/4000 [==============================] - 0s 85us/step - loss: 0.0315 - acc: 0.9955 - val_lo
Epoch 16/50
4000/4000 [==============================] - 0s 82us/step - loss: 0.0211 - acc: 0.9982 - val_lo
Epoch 17/50
4000/4000 [==============================] - 0s 82us/step - loss: 0.0320 - acc: 0.9925 - val_lo
Epoch 18/50
4000/4000 [==============================] - 0s 88us/step - loss: 0.0781 - acc: 0.9740 - val_lo
Epoch 19/50
4000/4000 [==============================] - 0s 96us/step - loss: 0.1338 - acc: 0.9527 - val_lo
Epoch 20/50
4000/4000 [==============================] - 0s 91us/step - loss: 0.1403 - acc: 0.9483 - val_lo
Epoch 21/50
4000/4000 [==============================] - 0s 85us/step - loss: 0.0969 - acc: 0.9688 - val_lo
Epoch 22/50
4000/4000 [==============================] - 0s 82us/step - loss: 0.0407 - acc: 0.9888 - val_lo
Epoch 23/50
4000/4000 [==============================] - 0s 91us/step - loss: 0.0389 - acc: 0.9895 - val_lo
Epoch 24/50
4000/4000 [==============================] - 0s 91us/step - loss: 0.0174 - acc: 0.9963 - val_lo
Epoch 25/50
4000/4000 [==============================] - 0s 84us/step - loss: 0.0081 - acc: 0.9988 - val_lo
Epoch 26/50
4000/4000 [==============================] - 0s 84us/step - loss: 0.0025 - acc: 1.0000 - val_lo
Epoch 27/50
4000/4000 [==============================] - 0s 83us/step - loss: 0.0014 - acc: 1.0000 - val_lo
Epoch 28/50
4000/4000 [==============================] - 0s 91us/step - loss: 0.0011 - acc: 1.0000 - val_lo
Epoch 29/50
4000/4000 [==============================] - 0s 90us/step - loss: 9.0256e-04 - acc: 1.0000 - va
Epoch 30/50
4000/4000 [==============================] - 0s 87us/step - loss: 7.8576e-04 - acc: 1.0000 - va
Epoch 31/50
4000/4000 [==============================] - 0s 95us/step - loss: 6.9064e-04 - acc: 1.0000 - va
```

```
Epoch 32/50
4000/4000 [==============================] - 0s 90us/step - loss: 6.1655e-04 - acc: 1.0000 - va
Epoch 33/50
4000/4000 [==============================] - 0s 92us/step - loss: 5.5558e-04 - acc: 1.0000 - va
Epoch 34/50
4000/4000 [==============================] - 0s 86us/step - loss: 5.0239e-04 - acc: 1.0000 - va
Epoch 35/50
4000/4000 [==============================] - 0s 95us/step - loss: 4.5934e-04 - acc: 1.0000 - va
Epoch 36/50
4000/4000 [==============================] - 0s 95us/step - loss: 4.2123e-04 - acc: 1.0000 - va
Epoch 37/50
4000/4000 [==============================] - 0s 98us/step - loss: 3.8703e-04 - acc: 1.0000 - va
Epoch 38/50
4000/4000 [==============================] - 0s 94us/step - loss: 3.5936e-04 - acc: 1.0000 - va
Epoch 39/50
4000/4000 [==============================] - 0s 99us/step - loss: 3.3367e-04 - acc: 1.0000 - va
Epoch 40/50
4000/4000 [==============================] - 0s 97us/step - loss: 3.0978e-04 - acc: 1.0000 - va
Epoch 41/50
4000/4000 [==============================] - 0s 87us/step - loss: 2.8999e-04 - acc: 1.0000 - va
Epoch 42/50
4000/4000 [==============================] - 0s 91us/step - loss: 2.7190e-04 - acc: 1.0000 - va
Epoch 43/50
4000/4000 [==============================] - 0s 89us/step - loss: 2.5476e-04 - acc: 1.0000 - va
Epoch 44/50
4000/4000 [==============================] - 0s 89us/step - loss: 2.3919e-04 - acc: 1.0000 - va
Epoch 45/50
4000/4000 [==============================] - 0s 90us/step - loss: 2.2538e-04 - acc: 1.0000 - va
Epoch 46/50
4000/4000 [==============================] - 0s 99us/step - loss: 2.1189e-04 - acc: 1.0000 - va
Epoch 47/50
4000/4000 [==============================] - 0s 89us/step - loss: 1.9991e-04 - acc: 1.0000 - va
Epoch 48/50
4000/4000 [==============================] - 0s 96us/step - loss: 1.8886e-04 - acc: 1.0000 - va
Epoch 49/50
4000/4000 [==============================] - 0s 95us/step - loss: 1.7833e-04 - acc: 1.0000 - va
Epoch 50/50
4000/4000 [==============================] - 0s 94us/step - loss: 1.6866e-04 - acc: 1.0000 - va
```

```
In [433]: plt.plot(history.history['acc'], label='Train accuracy')
          plt.plot(history.history['val_acc'], label='Train accuracy')
          plt.legend()
```

```
Out[433]: <matplotlib.legend.Legend at 0x7f8dcf7917f0>
```



```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```