dl-completely-inelastic-collision-rnn

January 9, 2020

0.1 Deep Learning Completely Inelastic Collision Solver

This notebook implements a recurrent neural network to solve for the post collision velocity of a two-mass system where the collision is completely inelastic and the second mass is initially at rest. The model is formulated using a recurrent neural network (RNN) and its input is represented as a text string giving the masses and the initial velocity of the first mass. Exact solutions calculated using conservation of momentum equations are used to train the RNN and evaluate the accuracy of predictions. This notebook is inspired by the addition_rnn.py example included with Keras.

0.1.1 Import useful packaged including TensorFlow 2.0 and Keras.

The notebook utilizes tensorflow >= 2.0, which now includes keras, a package of high level wrappers designed to make building and training deep learning models easier.

[1]:
```
# import packages
import tensorflow as tf
import tensorflow.keras as keras
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D

from tensorflow.keras.layers import RNN, LSTM, TimeDistributed, RepeatVector,␣
 ↪Dense, LSTMCell, Dropout
# LSTM doesn't work well since cuDNN is compiled with certain restrictions.
# So, here I will create LSTM layers by wrapping LSTMCell in RNN
from tensorflow.keras.models import Sequential
from tensorflow.keras import optimizers
```

0.1.2 Check to see if a Tensorflow is installed with GPU support and if a GPU is available.

[2]:
```
if not tf.test.is_gpu_available():
    print('No GPU found. Training will be slower.')
else:
    print('Default GPU {} found.'.format(tf.test.gpu_device_name()))
```

```
Default GPU /device:GPU:0 found.
```

0.1.3 Define a class to encode and decode between a selection of characters and one-hot integer representations.

```
[3]: class CharacterTable(object):
    """Given a set of characters:
    + Encode them to a one-hot integer representation
    + Decode the one-hot or integer representation to their character output
    + Decode a vector of probabilities to their character output
    """
    def __init__(self, chars):
        """Initialize character table.
        # Arguments
            chars: Characters that can appear in the input.
        """
        self.chars = sorted(set(chars))
        self.char_indices = dict((c, i) for i, c in enumerate(self.chars))
        self.indices_char = dict((i, c) for i, c in enumerate(self.chars))

    def encode(self, C, num_rows):
        """One-hot encode given string C.
        # Arguments
            C: string, to be encoded.
            num_rows: Number of rows in the returned one-hot encoding. This is
                used to keep the # of rows for each data the same.
        """
        x = np.zeros((num_rows, len(self.chars)))
        for i, c in enumerate(C):
            x[i, self.char_indices[c]] = 1
        return x

    def decode(self, x, calc_argmax=True):
        """Decode the given vector or 2D array to their character output.
        # Arguments
            x: A vector or a 2D array of probabilities or one-hot␣
↪representations;
                or a vector of character indices (used with␣
↪`calc_argmax=False`).
            calc_argmax: Whether to find the character index with maximum
                probability, defaults to `True`.
        """
        if calc_argmax:
            x = x.argmax(axis=-1)
        return ''.join(self.indices_char[x] for x in x)
```

```
class colors:
    ok = '\033[92m'
    fail = '\033[91m'
    close = '\033[0m'
```

0.1.4 Generate input and output character sequences.

The input is in the form {}m{}v{} where the values in the {}s are randomly drawn one or two character sequences representing, from left to right, the mass of object one, the mass of object two, and the initial velocity of object one. Object two is initially at rest.

The output is a character string representing the post collision velocity of the system after a head on totally inelastic collision.

[4]:
```
# Parameters for the model and dataset.
num_problems = 80000 # number of sequences in the training and validation sets
digits = 2 # maximum number of digits for mass and velocity in the input␣
 ↪sequence
lenans = 4 # number of characters in the answer sequence

REVERSE=False

# Maximum length of input is 'int + int' (e.g., '345+678'). Maximum length of
# int is DIGITS.
maxlen = digits + 1 + digits + 1 + digits

# All the numbers, plus sign and space for padding.
chars = '0123456789mv. '
ctable = CharacterTable(chars)

questions = []
expected = []
seen = set()
print('Generating data...')
while len(questions) < num_problems:
    f = lambda: int(''.join(np.random.choice(list('123456789'))
                    for i in np.arange(np.random.randint(1, digits + 1))))
    a, b, c = f(), f(), f()
    # Skip any questions we've already seen
    key = tuple(sorted((a, b, c)))
    if key in seen:
        continue
    seen.add(key)
    # Pad the data with spaces such that it is always MAXLEN.
    q = '{}m{}v{}'.format(a, b, c)
    query = q + ' ' * (maxlen - len(q))
    ans = a*c/(a+b)
```

```
    if ans < 10:
        r = lenans-2
    elif ans < 100:
        r = lenans-3
    ans = str(round(a*c/(a+b),r))
    # Answers can be of maximum size LENANS.
    ans += '0' * (lenans - len(ans))
    questions.append(query)
    expected.append(ans)
print('Total momentum questions:', len(questions))

print('Vectorization...')
x = np.zeros((len(questions), maxlen, len(chars)), dtype=np.bool)
y = np.zeros((len(questions), lenans, len(chars)), dtype=np.bool)
for i, sentence in enumerate(questions):
    x[i] = ctable.encode(sentence, maxlen)
for i, sentence in enumerate(expected):
    y[i] = ctable.encode(sentence, lenans)

# Shuffle (x, y) in unison as the later parts of x will almost all be larger
# digits.
indices = np.arange(len(y))
np.random.shuffle(indices)
x = x[indices]
y = y[indices]

# set apart 20% for validation and text data that we never train over.
split_at = len(x) - len(x) // 5
(x_train, x_val) = x[:split_at], x[split_at:]
(y_train, y_val) = y[:split_at], y[split_at:]

split_at =  len(x_val) - len(x_val) // 2
(x_val, x_test) = x_val[:split_at], x_val[split_at:]
(y_val, y_test) = y_val[:split_at], y_val[split_at:]

print('Training Data:')
print(x_train.shape)
print(y_train.shape)

print('Validation Data:')
print(x_val.shape)
print(y_val.shape)

print('Test Data:')
print(x_test.shape)
print(y_test.shape)
```

```
Generating data…
Total momentum questions: 80000
Vectorization…
Training Data:
(64000, 8, 14)
(64000, 4, 14)
Validation Data:
(8000, 8, 14)
(8000, 4, 14)
Test Data:
(8000, 8, 14)
(8000, 4, 14)
```

[5]:
```
# show an example of the input and output character strings
ii = np.random.randint(0, 1000)
print('input:', questions[ii], 'output:', expected[ii])
```

```
input: 1m9v93   output: 9.30
```

[6]:
```
# hyperparameters
HIDDEN_SIZE = 128
BATCH_SIZE = 256
LAYERS = 2
learning_rate = 0.001

print('Build model...')
model = Sequential()
# "encode" the input sequence using an RNN, producing an output of HIDDEN_SIZE.
# note: in a situation where your input sequences have a variable length, use␣
↪input_shape=(None, num_feature).
# model.add(LSTM(HIDDEN_SIZE, input_shape=(MAXLEN, len(chars))))
model.add(RNN(LSTMCell(HIDDEN_SIZE), input_shape=(maxlen, len(chars))))

# as the decoder RNN's input, repeatedly provide with the last output of
# RNN for each time step. Repeat 'lenans' times as that's the maximum length of␣
↪output.
model.add(RepeatVector(lenans))
# the decoder RNN could be multiple layers stacked or a single layer.
for _ in range(LAYERS):
    # by setting return_sequences to True, return not only the last output but
    # all the outputs so far in the form of (num_samples, timesteps,
    # output_dim). This is necessary as TimeDistributed in the below expects
    # the first dimension to be the timesteps.
    # model.add(LSTM(HIDDEN_SIZE, return_sequences=True))
    model.add(RNN(LSTMCell(HIDDEN_SIZE), return_sequences=True))

# add a dropout layer to prevent overfitting
```

```
# model.add(Dropout(rate=0.1))
# add a dense layer to every temporal slice of an input. for each of step of␣
↪the output sequence,
# decide which character should be chosen.
model.add(TimeDistributed(Dense(len(chars), activation='softmax')))
model.compile(loss='categorical_crossentropy',
              # optimizer=keras.optimizers.Adam(lr=learning_rate),
              optomizer=keras.optimizers.Adam(),
              metrics=['accuracy'])
model.summary()
```

```
Build model…
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rnn (RNN)                    (None, 128)               73216
_________________________________________________________________
repeat_vector (RepeatVector) (None, 4, 128)            0
_________________________________________________________________
rnn_1 (RNN)                  (None, 4, 128)            131584
_________________________________________________________________
rnn_2 (RNN)                  (None, 4, 128)            131584
_________________________________________________________________
time_distributed (TimeDistri (None, 4, 14)             1806
=================================================================
Total params: 338,190
Trainable params: 338,190
Non-trainable params: 0
_________________________________________________________________
```

0.1.5 Train the model and generate example predictions every 3 epochs.

[7]:
```
# train the model and show predictions against the validation dataset.
for iteration in range(1, 36):
    print()
    print('-' * 50)
    print('Iteration', iteration)
    model.fit(x_train, y_train,
              batch_size=BATCH_SIZE,
              epochs=3,
              validation_data=(x_val, y_val))
    score = model.evaluate(x_test, y_test, verbose=0)
    print('score:', score)
    # select 10 samples from the validation set at random so we can visualize␣
↪errors.
```

```python
    for i in range(10):
        ind = np.random.randint(0, len(x_val))
        rowx, rowy = x_test[np.array([ind])], y_test[np.array([ind])]
        preds = model.predict_classes(1.0*rowx, verbose=0) # mult by 1.0 to get␣
↪data types to jive
        q = ctable.decode(rowx[0])
        correct = ctable.decode(rowy[0])
        guess = ctable.decode(preds[0], calc_argmax=False)
        print('Q', q[::-1] if REVERSE else q, end=' ')
        print('A', correct, end=' ')

        for ii in np.arange(len(correct)):
            if correct[ii] == guess[ii]:
                print(colors.ok + ' ' + colors.close, end='')
            else:
                print(colors.fail + ' ' + colors.close, end='')

        print(' P', guess)
```

```
--------------------------------------------------
Iteration 1
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 6s 91us/sample - loss: 2.0252 -
accuracy: 0.3221 - val_loss: 1.9072 - val_accuracy: 0.3399
Epoch 2/3
64000/64000 [==============================] - 2s 39us/sample - loss: 1.8240 -
accuracy: 0.3581 - val_loss: 1.6968 - val_accuracy: 0.3859
Epoch 3/3
64000/64000 [==============================] - 2s 39us/sample - loss: 1.6521 -
accuracy: 0.3887 - val_loss: 1.5840 - val_accuracy: 0.3912
score: [1.5855826959609984, 0.39]
Q 3m92v97  A 3.06    P 1.30
Q 94m53v88 A 56.3    P 44.6
Q 74m74v4  A 2.00    P 1.38
Q 29m32v31 A 14.7    P 11.3
Q 82m7v32  A 29.5    P 24.8
Q 59m98v4  A 1.50    P 0.33
Q 4m89v74  A 3.18    P 1.38
Q 77m16v75 A 62.1    P 54.6
Q 81m77v49 A 25.1    P 14.3
Q 41m9v74  A 60.7    P 58.8

--------------------------------------------------
Iteration 2
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 3s 40us/sample - loss: 1.5172 -
accuracy: 0.4214 - val_loss: 1.4270 - val_accuracy: 0.4591
Epoch 2/3
64000/64000 [==============================] - 2s 39us/sample - loss: 1.4187 -
accuracy: 0.4543 - val_loss: 1.4477 - val_accuracy: 0.4371
Epoch 3/3
64000/64000 [==============================] - 2s 39us/sample - loss: 1.3556 -
accuracy: 0.4751 - val_loss: 1.3870 - val_accuracy: 0.4625
score: [1.38504066324234, 0.46590626]
Q 29m2v13  A 12.2    P 10.8
Q 58m76v27 A 11.7    P 13.8
Q 28m15v41 A 26.7    P 20.8
Q 16m8v61  A 40.7    P 48.0
Q 65m57v31 A 16.5    P 18.8
Q 5m27v97  A 15.2    P 15.8
Q 16m8v22  A 14.7    P 17.8
Q 68m5v56  A 52.2    P 50.8
Q 18m59v42 A 9.82    P 12.8
Q 61m85v66 A 27.6    P 30.8

--------------------------------------------------
Iteration 3
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 3s 40us/sample - loss: 1.3155 -
accuracy: 0.4893 - val_loss: 1.3143 - val_accuracy: 0.4773
Epoch 2/3
64000/64000 [==============================] - 3s 40us/sample - loss: 1.2835 -
accuracy: 0.5013 - val_loss: 1.2968 - val_accuracy: 0.4835
Epoch 3/3
64000/64000 [==============================] - 3s 40us/sample - loss: 1.2592 -
accuracy: 0.5097 - val_loss: 1.2515 - val_accuracy: 0.5061
score: [1.25640402507782, 0.5069375]
Q 52m76v4  A 1.62    P 1.80
Q 26m31v82 A 37.4    P 44.2
Q 23m25v14 A 6.71    P 6.80
Q 43m34v8  A 4.47    P 4.20
Q 78m48v1  A 0.62    P 0.62
Q 92m1v15  A 14.8    P 16.8
Q 89m86v44 A 22.4    P 24.2
Q 5m57v6   A 0.48    P 0.42
Q 4m81v7   A 0.33    P 0.42
Q 24m24v23 A 11.5    P 12.2

--------------------------------------------------
Iteration 4
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 3s 40us/sample - loss: 1.2372 -
accuracy: 0.5167 - val_loss: 1.2266 - val_accuracy: 0.5182
Epoch 2/3
64000/64000 [==============================] - 3s 40us/sample - loss: 1.2172 -
accuracy: 0.5231 - val_loss: 1.2430 - val_accuracy: 0.5013
Epoch 3/3
64000/64000 [==============================] - 3s 40us/sample - loss: 1.1993 -
accuracy: 0.5298 - val_loss: 1.2064 - val_accuracy: 0.5277
score: [1.2089817209243774, 0.526875]
Q 63m58v25 A 13.0    P 12.2
Q 37m58v34 A 13.2    P 13.2
Q 45m75v36 A 13.5    P 13.2
Q 47m91v45 A 15.3    P 14.2
Q 45m4v23  A 21.1    P 21.6
Q 84m54v21 A 12.8    P 12.2
Q 62m42v81 A 48.3    P 49.6
Q 81m41v23 A 15.3    P 14.2
Q 9m57v27  A 3.68    P 3.70
Q 55m68v87 A 38.9    P 35.7

--------------------------------------------------
Iteration 5
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 3s 40us/sample - loss: 1.1831 -
accuracy: 0.5356 - val_loss: 1.1539 - val_accuracy: 0.5506
Epoch 2/3
64000/64000 [==============================] - 3s 40us/sample - loss: 1.1664 -
accuracy: 0.5411 - val_loss: 1.1241 - val_accuracy: 0.5673
Epoch 3/3
64000/64000 [==============================] - 3s 40us/sample - loss: 1.1503 -
accuracy: 0.5472 - val_loss: 1.1327 - val_accuracy: 0.5453
score: [1.1363491096496583, 0.54075]
Q 38m85v21 A 6.49    P 6.11
Q 78m59v97 A 55.2    P 56.5
Q 99m5v67  A 63.8    P 61.1
Q 21m31v8  A 3.23    P 3.50
Q 33m87v16 A 4.40    P 4.80
Q 16m71v22 A 4.05    P 4.11
Q 16m8v22  A 14.7    P 13.4
Q 89m3v82  A 79.3    P 78.1
Q 5m85v12  A 0.67    P 0.75
Q 88m24v29 A 22.8    P 21.8

--------------------------------------------------
Iteration 6
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 3s 41us/sample - loss: 1.1355 -
accuracy: 0.5518 - val_loss: 1.1521 - val_accuracy: 0.5374
Epoch 2/3
64000/64000 [==============================] - 3s 40us/sample - loss: 1.1223 -
accuracy: 0.5569 - val_loss: 1.0885 - val_accuracy: 0.5785
Epoch 3/3
64000/64000 [==============================] - 3s 40us/sample - loss: 1.1066 -
accuracy: 0.5634 - val_loss: 1.1267 - val_accuracy: 0.5455
score: [1.1247176299095154, 0.545625]
Q 82m19v44 A 35.7    P 37.3
Q 41m13v78 A 59.2    P 61.0
Q 31m35v31 A 14.6    P 15.3
Q 39m54v21 A 8.81    P 8.10
Q 75m58v85 A 47.9    P 50.3
Q 21m8v18  A 13.0    P 14.8
Q 75m3v92  A 88.5    P 89.0
Q 83m1v13  A 12.8    P 13.8
Q 84m73v32 A 17.1    P 17.3
Q 48m7v58  A 50.6    P 51.8

--------------------------------------------------
Iteration 7
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 3s 40us/sample - loss: 1.0948 -
accuracy: 0.5667 - val_loss: 1.0560 - val_accuracy: 0.5927
Epoch 2/3
64000/64000 [==============================] - 3s 41us/sample - loss: 1.0814 -
accuracy: 0.5710 - val_loss: 1.0823 - val_accuracy: 0.5720
Epoch 3/3
64000/64000 [==============================] - 3s 40us/sample - loss: 1.0692 -
accuracy: 0.5775 - val_loss: 1.0657 - val_accuracy: 0.5742
score: [1.0661763501167298, 0.5701875]
Q 14m39v26 A 6.87    P 6.60
Q 38m58v28 A 11.1    P 11.1
Q 92m97v29 A 14.1    P 13.1
Q 59m36v4  A 2.48    P 2.10
Q 98m76v38 A 21.4    P 20.1
Q 1m11v45  A 3.75    P 3.70
Q 8m23v35  A 9.03    P 7.00
Q 14m89v97 A 13.2    P 13.1
Q 22m54v78 A 22.6    P 22.3
Q 98m88v24 A 12.6    P 12.1

--------------------------------------------------
Iteration 8
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 2s 38us/sample - loss: 1.0580 -
accuracy: 0.5813 - val_loss: 1.1565 - val_accuracy: 0.5343
Epoch 2/3
64000/64000 [==============================] - 2s 38us/sample - loss: 1.0475 -
accuracy: 0.5848 - val_loss: 1.0371 - val_accuracy: 0.5832
Epoch 3/3
64000/64000 [==============================] - 2s 38us/sample - loss: 1.0366 -
accuracy: 0.5898 - val_loss: 1.0094 - val_accuracy: 0.5998
score: [1.0074513502120972, 0.6014063]
Q 99m23v56 A 45.4    P 44.5
Q 61m2v94  A 91.0    P 91.1
Q 46m97v44 A 14.2    P 13.9
Q 64m89v62 A 25.9    P 25.5
Q 91m1v27  A 26.7    P 26.6
Q 8m18v43  A 13.2    P 12.0
Q 38m2v71  A 67.5    P 68.9
Q 22m8v73  A 53.5    P 54.5
Q 1m51v22  A 0.42    P 0.25
Q 41m34v63 A 34.4    P 33.5

--------------------------------------------------
Iteration 9
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 2s 38us/sample - loss: 1.0279 -
accuracy: 0.5941 - val_loss: 1.0148 - val_accuracy: 0.5924
Epoch 2/3
64000/64000 [==============================] - 2s 38us/sample - loss: 1.0197 -
accuracy: 0.5962 - val_loss: 1.0253 - val_accuracy: 0.5934
Epoch 3/3
64000/64000 [==============================] - 2s 38us/sample - loss: 1.0099 -
accuracy: 0.6014 - val_loss: 1.0736 - val_accuracy: 0.5699
score: [1.0733536467552185, 0.568375]
Q 2m25v31  A 2.30    P 2.90
Q 41m76v58 A 20.3    P 20.8
Q 41m13v78 A 59.2    P 69.9
Q 69m66v62 A 31.7    P 32.9
Q 8m48v61  A 8.71    P 8.00
Q 67m6v62  A 56.9    P 58.1
Q 24m72v15 A 3.75    P 3.12
Q 61m7v13  A 11.7    P 12.4
Q 94m43v89 A 61.1    P 61.9
Q 41m59v11 A 4.51    P 4.12

--------------------------------------------------
Iteration 10
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 2s 39us/sample - loss: 1.0022 -
accuracy: 0.6037 - val_loss: 0.9463 - val_accuracy: 0.6365
Epoch 2/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.9947 -
accuracy: 0.6069 - val_loss: 0.9755 - val_accuracy: 0.6210
Epoch 3/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.9860 -
accuracy: 0.6104 - val_loss: 0.9564 - val_accuracy: 0.6224
score: [0.9550179929733277, 0.6228125]
Q 41m34v63 A 34.4    P 33.5
Q 24m48v81 A 27.0    P 26.6
Q 36m37v7  A 3.45    P 3.60
Q 96m64v86 A 51.6    P 50.6
Q 99m18v29 A 24.5    P 23.6
Q 15m75v73 A 12.2    P 12.6
Q 13m75v62 A 9.16    P 9..2
Q 7m89v7   A 0.51    P 0.42
Q 91m26v1  A 0.78    P 0.76
Q 49m98v9  A 3.00    P 2.56

--------------------------------------------------
Iteration 11
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.9791 -
accuracy: 0.6124 - val_loss: 0.9770 - val_accuracy: 0.6109
Epoch 2/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.9716 -
accuracy: 0.6165 - val_loss: 1.0375 - val_accuracy: 0.5902
Epoch 3/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.9655 -
accuracy: 0.6185 - val_loss: 0.9217 - val_accuracy: 0.6428
score: [0.9180031900405884, 0.6445625]
Q 48m17v79 A 58.3    P 58.8
Q 84m15v47 A 39.9    P 49.8
Q 13m37v59 A 15.3    P 15.1
Q 61m59v12 A 6.10    P 6.10
Q 68m16v33 A 26.7    P 26.1
Q 58m13v95 A 77.6    P 76.1
Q 39m97v91 A 26.1    P 25.1
Q 9m17v49  A 17.0    P 16.8
Q 77m49v3  A 1.83    P 1.78
Q 56m44v28 A 15.7    P 15.1

--------------------------------------------------
Iteration 12
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.9587 -
accuracy: 0.6226 - val_loss: 0.9403 - val_accuracy: 0.6226
Epoch 2/3
64000/64000 [==============================] - 3s 40us/sample - loss: 0.9515 -
accuracy: 0.6248 - val_loss: 1.0730 - val_accuracy: 0.5728
Epoch 3/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.9478 -
accuracy: 0.6260 - val_loss: 1.0166 - val_accuracy: 0.6011
score: [1.0162253673076629, 0.601375]
Q 86m99v32 A 14.9    P 15.8
Q 89m86v44 A 22.4    P 22.8
Q 35m6v39  A 33.3    P 33.8
Q 85m51v63 A 39.4    P 38.0
Q 46m65v77 A 31.9    P 32.8
Q 83m7v45  A 41.5    P 41.8
Q 41m2v99  A 94.4    P 94.8
Q 26m52v85 A 28.3    P 28.2
Q 16m57v4  A 0.88    P 0.94
Q 73m19v16 A 12.7    P 12.8

--------------------------------------------------
Iteration 13
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 3s 40us/sample - loss: 0.9416 -
accuracy: 0.6291 - val_loss: 0.9205 - val_accuracy: 0.6395
Epoch 2/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.9360 -
accuracy: 0.6313 - val_loss: 0.8947 - val_accuracy: 0.6570
Epoch 3/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.9322 -
accuracy: 0.6318 - val_loss: 0.8813 - val_accuracy: 0.6644
score: [0.8812202425003052, 0.66353124]
Q 93m8v11  A 10.1    P 10.1
Q 73m28v2  A 1.45    P 1.43
Q 24m83v43 A 9.64    P 9..0
Q 13m75v56 A 8.27    P 8.90
Q 13m91v78 A 9.75    P 9..1
Q 52m9v1   A 0.85    P 0.80
Q 33m6v64  A 54.2    P 54.0
Q 58m55v63 A 32.3    P 32.0
Q 57m93v74 A 28.1    P 28.0
Q 77m18v43 A 34.9    P 35.0

--------------------------------------------------
Iteration 14
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 3s 40us/sample - loss: 0.9266 -
accuracy: 0.6358 - val_loss: 0.9004 - val_accuracy: 0.6448
Epoch 2/3
64000/64000 [==============================] - 3s 40us/sample - loss: 0.9201 -
accuracy: 0.6375 - val_loss: 0.9676 - val_accuracy: 0.6114
Epoch 3/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.9159 -
accuracy: 0.6390 - val_loss: 0.9426 - val_accuracy: 0.6218
score: [0.9460811560153961, 0.61928123]
Q 75m6v58  A 53.7    P 53.2
Q 31m58v62 A 21.6    P 21.2
Q 3m72v15  A 0.60    P 0.62
Q 97m65v73 A 43.7    P 42.2
Q 25m74v55 A 13.9    P 13.2
Q 41m2v99  A 94.4    P 94.0
Q 24m42v92 A 33.5    P 33.2
Q 12m79v69 A 9.10    P 8.22
Q 8m21v72  A 19.9    P 29.2
Q 23m11v85 A 57.5    P 56.2

--------------------------------------------------
Iteration 15
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.9127 -
accuracy: 0.6403 - val_loss: 0.9325 - val_accuracy: 0.6245
Epoch 2/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.9073 -
accuracy: 0.6424 - val_loss: 0.8621 - val_accuracy: 0.6674
Epoch 3/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.9047 -
accuracy: 0.6437 - val_loss: 0.8590 - val_accuracy: 0.6737
score: [0.8556039447784424, 0.6741875]
Q 56m49v91 A 48.5    P 48.0
Q 62m75v51 A 23.1    P 23.6
Q 4m98v18  A 0.71    P 0.72
Q 3m81v14  A 0.50    P 0.53
Q 61m91v2  A 0.80    P 0.86
Q 22m88v1  A 0.20    P 0.22
Q 8m33v85  A 16.6    P 15.2
Q 8m71v15  A 1.52    P 1.53
Q 61m53v6  A 3.21    P 3.10
Q 92m25v38 A 29.9    P 29.6

--------------------------------------------------
Iteration 16
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.8982 -
accuracy: 0.6468 - val_loss: 0.9359 - val_accuracy: 0.6243
Epoch 2/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.8949 -
accuracy: 0.6485 - val_loss: 0.8677 - val_accuracy: 0.6587
Epoch 3/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.8915 -
accuracy: 0.6490 - val_loss: 0.9218 - val_accuracy: 0.6384
score: [0.919591515302658, 0.6401875]
Q 6m91v58  A 3.59    P 3.00
Q 9m9v55   A 27.5    P 27.0
Q 48m56v72 A 33.2    P 32.0
Q 27m32v94 A 43.0    P 42.0
Q 49m74v24 A 9.56    P 9.40
Q 71m37v57 A 37.5    P 37.0
Q 17m57v29 A 6.66    P 6.00
Q 19m62v33 A 7.74    P 7.60
Q 11m45v22 A 4.32    P 4.70
Q 28m91v23 A 5.41    P 5.02

--------------------------------------------------
Iteration 17
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.8881 -
accuracy: 0.6504 - val_loss: 0.8694 - val_accuracy: 0.6616
Epoch 2/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.8826 -
accuracy: 0.6533 - val_loss: 0.8683 - val_accuracy: 0.6633
Epoch 3/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.8805 -
accuracy: 0.6540 - val_loss: 0.8797 - val_accuracy: 0.6520
score: [0.879545749425888, 0.6511875]
Q 26m42v64 A 24.5    P 24.5
Q 7m67v5   A 0.47    P 0.48
Q 52m82v24 A 9.31    P 9.60
Q 94m47v36 A 24.0    P 23.8
Q 8m37v43  A 7.64    P 7.50
Q 71m89v99 A 43.9    P 44.5
Q 53m59v28 A 13.2    P 13.5
Q 15m42v2  A 0.53    P 0.40
Q 43m14v25 A 18.9    P 19.5
Q 7m87v98  A 7.30    P 6.50

--------------------------------------------------
Iteration 18
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.8775 -
accuracy: 0.6547 - val_loss: 0.8412 - val_accuracy: 0.6785
Epoch 2/3
64000/64000 [==============================] - 3s 40us/sample - loss: 0.8729 -
accuracy: 0.6578 - val_loss: 0.9125 - val_accuracy: 0.6407
Epoch 3/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.8697 -
accuracy: 0.6592 - val_loss: 0.9355 - val_accuracy: 0.6315
score: [0.9333925821781158, 0.631375]
Q 32m8v9   A 7.20    P 6.58
Q 29m26v86 A 45.3    P 44.9
Q 5m24v24  A 4.14    P 3.55
Q 6m12v8   A 2.67    P 2.80
Q 96m95v8  A 4.02    P 4.90
Q 35m33v65 A 33.5    P 33.5
Q 98m77v6  A 3.36    P 3.51
Q 53m75v32 A 13.2    P 13.5
Q 35m5v4   A 3.50    P 3.47
Q 72m52v41 A 23.8    P 24.8

--------------------------------------------------
Iteration 19
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.8688 -
accuracy: 0.6578 - val_loss: 0.8574 - val_accuracy: 0.6658
Epoch 2/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.8644 -
accuracy: 0.6605 - val_loss: 0.8219 - val_accuracy: 0.6862
Epoch 3/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.8597 -
accuracy: 0.6615 - val_loss: 0.8497 - val_accuracy: 0.6683
score: [0.8487990510463714, 0.66809374]
Q 81m48v35 A 22.0    P 22.8
Q 75m96v11 A 4.82    P 4.88
Q 28m75v33 A 8.97    P 9.10
Q 2m65v67  A 2.00    P 2.08
Q 75m6v29  A 26.9    P 26.8
Q 16m83v99 A 16.0    P 16.8
Q 19m42v23 A 7.16    P 7.80
Q 9m11v32  A 14.4    P 14.0
Q 97m23v61 A 49.3    P 48.8
Q 7m89v94  A 6.85    P 6.88

--------------------------------------------------
Iteration 20
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.8561 -
accuracy: 0.6644 - val_loss: 0.9543 - val_accuracy: 0.6270
Epoch 2/3
64000/64000 [==============================] - 3s 40us/sample - loss: 0.8556 -
accuracy: 0.6645 - val_loss: 0.8521 - val_accuracy: 0.6663
Epoch 3/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.8514 -
accuracy: 0.6655 - val_loss: 0.8926 - val_accuracy: 0.6453
score: [0.8878784575462342, 0.64821875]
Q 57m72v64 A 28.3    P 28.5
Q 34m92v85 A 22.9    P 23.1
Q 85m51v63 A 39.4    P 39.5
Q 26m54v5  A 1.62    P 1.65
Q 92m2v39  A 38.2    P 38.1
Q 9m7v46   A 25.9    P 26.5
Q 65m3v68  A 65.0    P 65.2
Q 76m8v51  A 46.1    P 46.5
Q 74m36v83 A 55.8    P 55.1
Q 36m59v22 A 8.34    P 8.60

--------------------------------------------------
Iteration 21
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.8488 -
accuracy: 0.6674 - val_loss: 0.8952 - val_accuracy: 0.6420
Epoch 2/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.8433 -
accuracy: 0.6698 - val_loss: 0.8314 - val_accuracy: 0.6712
Epoch 3/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.8416 -
accuracy: 0.6701 - val_loss: 0.8651 - val_accuracy: 0.6618
score: [0.866441558599472, 0.6590313]
Q 22m54v57 A 16.5    P 16.0
Q 8m35v13  A 2.42    P 2.20
Q 41m48v8  A 3.69    P 3.50
Q 52m74v12 A 4.95    P 4.30
Q 78m46v29 A 18.2    P 18.8
Q 2m89v29  A 0.64    P 0.53
Q 28m41v73 A 29.6    P 29.3
Q 75m3v92  A 88.5    P 88.5
Q 37m81v79 A 24.8    P 24.3
Q 71m14v5  A 4.18    P 4.18

--------------------------------------------------
Iteration 22
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.8392 -
accuracy: 0.6709 - val_loss: 0.8508 - val_accuracy: 0.6631
Epoch 2/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.8361 -
accuracy: 0.6733 - val_loss: 0.8439 - val_accuracy: 0.6679
Epoch 3/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.8347 -
accuracy: 0.6732 - val_loss: 0.8815 - val_accuracy: 0.6482
score: [0.8830624108314514, 0.6481562]
Q 59m57v24 A 12.2    P 11.0
Q 53m6v8   A 7.19    P 7.25
Q 7m23v17  A 3.97    P 3.00
Q 47m18v98 A 70.9    P 71.0
Q 28m88v8  A 1.93    P 1.80
Q 98m91v18 A 9.33    P 9.10
Q 55m97v86 A 31.1    P 30.3
Q 24m9v17  A 12.4    P 12.1
Q 57m75v51 A 22.0    P 21.1
Q 43m21v7  A 4.70    P 4.10

--------------------------------------------------
Iteration 23
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.8305 -
accuracy: 0.6741 - val_loss: 0.8411 - val_accuracy: 0.6668
Epoch 2/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.8275 -
accuracy: 0.6763 - val_loss: 0.8227 - val_accuracy: 0.6740
Epoch 3/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.8267 -
accuracy: 0.6771 - val_loss: 0.8481 - val_accuracy: 0.6644
score: [0.8486135015487671, 0.6670625]
Q 2m53v65  A 2.36    P 2.20
Q 76m84v66 A 31.4    P 31.8
Q 22m13v41 A 25.8    P 25.0
Q 17m3v45  A 38.2    P 38.8
Q 97m94v66 A 33.5    P 33.0
Q 34m38v27 A 12.8    P 12.8
Q 6m8v59   A 25.3    P 24.0
Q 97m44v29 A 20.0    P 20.8
Q 5m95v11  A 0.55    P 0.43
Q 18m38v61 A 19.6    P 18.0

--------------------------------------------------
Iteration 24
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.8223 -
accuracy: 0.6785 - val_loss: 0.8180 - val_accuracy: 0.6837
Epoch 2/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.8195 -
accuracy: 0.6793 - val_loss: 0.8330 - val_accuracy: 0.6690
Epoch 3/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.8179 -
accuracy: 0.6807 - val_loss: 0.7950 - val_accuracy: 0.6927
score: [0.7929418609142304, 0.6969375]
Q 47m67v49 A 20.2    P 20.5
Q 76m8v51  A 46.1    P 46.8
Q 46m69v93 A 37.2    P 37.0
Q 38m51v35 A 14.9    P 15.6
Q 38m18v19 A 12.9    P 12.8
Q 2m22v77  A 6.42    P 6.00
Q 14m22v42 A 16.3    P 16.0
Q 81m66v21 A 11.6    P 11.5
Q 68m6v54  A 49.6    P 59.8
Q 47m9v66  A 55.4    P 55.5

--------------------------------------------------
Iteration 25
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.8155 -
accuracy: 0.6811 - val_loss: 0.7967 - val_accuracy: 0.6908
Epoch 2/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.8133 -
accuracy: 0.6815 - val_loss: 0.8581 - val_accuracy: 0.6598
Epoch 3/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.8105 -
accuracy: 0.6831 - val_loss: 0.8080 - val_accuracy: 0.6808
score: [0.8066634085178376, 0.68290627]
Q 95m54v58 A 37.0    P 37.5
Q 95m16v35 A 30.0    P 30.8
Q 5m24v63  A 10.9    P 11.5
Q 81m8v16  A 14.6    P 14.6
Q 4m37v53  A 5.17    P 5.00
Q 61m34v52 A 33.4    P 33.5
Q 71m85v51 A 23.2    P 23.5
Q 92m8v78  A 71.8    P 71.5
Q 12m53v9  A 1.66    P 1.55
Q 94m86v39 A 20.4    P 20.5

--------------------------------------------------
Iteration 26
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.8074 -
accuracy: 0.6841 - val_loss: 0.8086 - val_accuracy: 0.6776
Epoch 2/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.8052 -
accuracy: 0.6858 - val_loss: 0.8475 - val_accuracy: 0.6694
Epoch 3/3
64000/64000 [==============================] - 3s 40us/sample - loss: 0.8019 -
accuracy: 0.6864 - val_loss: 0.8037 - val_accuracy: 0.6816
score: [0.8009607315063476, 0.6852813]
Q 67m44v71 A 42.9    P 43.0
Q 96m37v56 A 40.4    P 40.8
Q 67m32v67 A 45.3    P 45.0
Q 47m98v7  A 2.27    P 2.33
Q 72m2v4   A 3.89    P 3.89
Q 57m23v52 A 37.0    P 37.0
Q 62m94v54 A 21.5    P 21.5
Q 6m76v76  A 5.56    P 5.50
Q 84m98v34 A 15.7    P 16.1
Q 12m96v29 A 3.22    P 3.30

--------------------------------------------------
Iteration 27
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.7995 -
accuracy: 0.6881 - val_loss: 0.8097 - val_accuracy: 0.6853
Epoch 2/3
64000/64000 [==============================] - 3s 40us/sample - loss: 0.7986 -
accuracy: 0.6888 - val_loss: 0.8176 - val_accuracy: 0.6776
Epoch 3/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.7955 -
accuracy: 0.6890 - val_loss: 0.8158 - val_accuracy: 0.6735
score: [0.8188097815513611, 0.67175]
Q 26m5v34  A 28.5    P 28.8
Q 62m26v86 A 60.6    P 60.9
Q 23m15v7  A 4.24    P 4.20
Q 98m4v41  A 39.4    P 39.4
Q 5m68v84  A 5.75    P 5.09
Q 37m27v56 A 32.4    P 32.9
Q 83m12v43 A 37.6    P 36.9
Q 24m72v59 A 14.8    P 15.2
Q 67m91v44 A 18.7    P 18.9
Q 53m59v28 A 13.2    P 13.6

--------------------------------------------------
Iteration 28
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.7933 -
accuracy: 0.6895 - val_loss: 0.8693 - val_accuracy: 0.6583
Epoch 2/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.7918 -
accuracy: 0.6907 - val_loss: 0.7983 - val_accuracy: 0.6832
Epoch 3/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.7881 -
accuracy: 0.6927 - val_loss: 0.8026 - val_accuracy: 0.6790
score: [0.8030582339763641, 0.6786875]
Q 25m29v65 A 30.1    P 30.6
Q 66m24v5  A 3.67    P 3.66
Q 51m96v93 A 32.3    P 32.0
Q 68m81v19 A 8.67    P 8.60
Q 7m22v43  A 10.4    P 10.6
Q 25m53v75 A 24.0    P 23.3
Q 35m35v8  A 4.00    P 3.90
Q 58m59v98 A 48.6    P 49.5
Q 37m47v85 A 37.4    P 37.0
Q 62m15v3  A 2.42    P 2.40

--------------------------------------------------
Iteration 29
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.7872 -
accuracy: 0.6931 - val_loss: 0.7769 - val_accuracy: 0.6988
Epoch 2/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.7840 -
accuracy: 0.6938 - val_loss: 0.8692 - val_accuracy: 0.6630
Epoch 3/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.7817 -
accuracy: 0.6953 - val_loss: 0.8320 - val_accuracy: 0.6703
score: [0.8350286138057709, 0.66809374]
Q 9m47v78  A 12.5    P 12.8
Q 16m19v1  A 0.46    P 0.46
Q 72m27v99 A 72.0    P 71.6
Q 34m38v27 A 12.8    P 12.8
Q 4m56v28  A 1.87    P 1.90
Q 54m58v39 A 18.8    P 18.5
Q 66m58v31 A 16.5    P 16.6
Q 48m6v2   A 1.78    P 1.75
Q 56m12v71 A 58.5    P 57.6
Q 94m84v61 A 32.2    P 31.0

--------------------------------------------------
Iteration 30
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.7793 -
accuracy: 0.6959 - val_loss: 0.7784 - val_accuracy: 0.6968
Epoch 2/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.7776 -
accuracy: 0.6969 - val_loss: 0.7880 - val_accuracy: 0.6925
Epoch 3/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.7744 -
accuracy: 0.6982 - val_loss: 0.8171 - val_accuracy: 0.6817
score: [0.8186441586017609, 0.67934376]
Q 84m2v74  A 72.3    P 72.2
Q 47m98v35 A 11.3    P 11.0
Q 28m9v36  A 27.2    P 27.0
Q 3m45v68  A 4.25    P 4.25
Q 7m67v5   A 0.47    P 0.44
Q 18m95v79 A 12.6    P 12.0
Q 53m78v82 A 33.2    P 32.5
Q 41m57v7  A 2.93    P 2.90
Q 2m33v3   A 0.17    P 0.15
Q 13m43v83 A 19.3    P 19.2

--------------------------------------------------
Iteration 31
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 3s 40us/sample - loss: 0.7739 -
accuracy: 0.6985 - val_loss: 0.7901 - val_accuracy: 0.6886
Epoch 2/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.7712 -
accuracy: 0.6992 - val_loss: 0.7590 - val_accuracy: 0.7062
Epoch 3/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.7711 -
accuracy: 0.6995 - val_loss: 0.7859 - val_accuracy: 0.6930
score: [0.7823098633289337, 0.698]
Q 56m7v86  A 76.4    P 76.1
Q 79m57v92 A 53.4    P 53.3
Q 52m32v25 A 15.5    P 15.5
Q 42m14v87 A 65.2    P 65.5
Q 4m92v28  A 1.17    P 1.15
Q 58m89v35 A 13.8    P 13.8
Q 81m88v63 A 30.2    P 30.3
Q 2m61v3   A 0.10    P 0.09
Q 24m79v78 A 18.2    P 18.5
Q 15m42v2  A 0.53    P 0.50

--------------------------------------------------
Iteration 32
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.7674 -
accuracy: 0.7008 - val_loss: 0.8750 - val_accuracy: 0.6531
Epoch 2/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.7644 -
accuracy: 0.7033 - val_loss: 0.8079 - val_accuracy: 0.6768
Epoch 3/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.7633 -
accuracy: 0.7032 - val_loss: 0.7945 - val_accuracy: 0.6870
score: [0.7964618723392487, 0.68703127]
Q 38m9v29  A 23.4    P 23.0
Q 93m5v58  A 55.0    P 55.1
Q 6m81v47  A 3.24    P 3.20
Q 24m12v82 A 54.7    P 55.9
Q 79m68v7  A 3.76    P 3.77
Q 16m83v9  A 1.45    P 1.45
Q 99m45v5  A 3.44    P 3.51
Q 93m36v46 A 33.2    P 33.2
Q 21m8v18  A 13.0    P 12.5
Q 26m38v18 A 7.31    P 7.80

--------------------------------------------------
Iteration 33
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.7609 -
accuracy: 0.7042 - val_loss: 0.8159 - val_accuracy: 0.6761
Epoch 2/3
64000/64000 [==============================] - 3s 39us/sample - loss: 0.7588 -
accuracy: 0.7043 - val_loss: 0.7773 - val_accuracy: 0.6978
Epoch 3/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.7567 -
accuracy: 0.7058 - val_loss: 0.8803 - val_accuracy: 0.6568
score: [0.8798099870681763, 0.65578127]
Q 67m23v24 A 17.9    P 17.8
Q 37m61v22 A 8.31    P 8.46
Q 3m83v7   A 0.24    P 0.24
Q 43m69v27 A 10.4    P 10.1
Q 34m79v98 A 29.5    P 29.2
Q 67m87v1  A 0.44    P 0.45
Q 47m44v21 A 10.8    P 10.8
Q 61m8v39  A 34.5    P 35.6
Q 35m33v28 A 14.4    P 14.0
Q 9m26v3   A 0.77    P 0.72

--------------------------------------------------
Iteration 34
Train on 64000 samples, validate on 8000 samples
```

```
Epoch 1/3
64000/64000 [==============================] - 3s 40us/sample - loss: 0.7554 -
accuracy: 0.7063 - val_loss: 0.7768 - val_accuracy: 0.6930
Epoch 2/3
64000/64000 [==============================] - 3s 40us/sample - loss: 0.7540 -
accuracy: 0.7064 - val_loss: 0.7572 - val_accuracy: 0.7030
Epoch 3/3
64000/64000 [==============================] - ETA: 0s - loss: 0.7515 -
accuracy: 0.70 - 2s 38us/sample - loss: 0.7521 - accuracy: 0.7061 - val_loss:
0.7697 - val_accuracy: 0.6945
score: [0.7694476416110992, 0.69578123]
Q 31m14v27 A 18.6    P 18.5
Q 57m26v17 A 11.7    P 11.6
Q 7m25v34  A 7.44    P 7.20
Q 25m6v91  A 73.4    P 74.8
Q 88m35v38 A 27.2    P 27.0
Q 47m2v98  A 94.0    P 94.2
Q 96m7v62  A 57.8    P 58.1
Q 88m7v91  A 84.3    P 84.3
Q 19m57v36 A 9.00    P 8.12
Q 68m17v3  A 2.40    P 2.41

--------------------------------------------------
Iteration 35
Train on 64000 samples, validate on 8000 samples
Epoch 1/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.7482 -
accuracy: 0.7089 - val_loss: 0.7972 - val_accuracy: 0.6853
Epoch 2/3
64000/64000 [==============================] - 2s 38us/sample - loss: 0.7480 -
accuracy: 0.7082 - val_loss: 0.7576 - val_accuracy: 0.7023
Epoch 3/3
64000/64000 [==============================] - 2s 39us/sample - loss: 0.7467 -
accuracy: 0.7091 - val_loss: 0.7545 - val_accuracy: 0.7038
score: [0.7542344851493835, 0.7033125]
Q 98m94v95 A 48.5    P 48.0
Q 36m62v35 A 12.9    P 12.6
Q 44m28v8  A 4.89    P 4.70
Q 55m8v53  A 46.3    P 46.5
Q 3m93v4   A 0.12    P 0.12
Q 9m48v83  A 13.1    P 12.4
Q 65m26v67 A 47.9    P 48.0
Q 92m72v59 A 33.1    P 32.0
Q 61m56v73 A 38.1    P 37.0
Q 9m58v42  A 5.64    P 5.40
```

[ ]: