In [1]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import keras
from keras.layers import Dense, Dropout, Flatten
from keras.models import Sequential
from keras import optimizers
from keras.optimizers import RMSprop, Adagrad, Adadelta
import matplotlib.pyplot as plt
```

```
Using TensorFlow backend.
C:\Users\pisar\Anaconda3\lib\site-packages\tensorflow\python\framework\dty
pes.py:516: FutureWarning: Passing (type, 1) or '1type' as a synonym of ty
pe is deprecated; in a future version of numpy, it will be understood as
(type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\pisar\Anaconda3\lib\site-packages\tensorflow\python\framework\dty
pes.py:517: FutureWarning: Passing (type, 1) or '1type' as a synonym of ty
pe is deprecated; in a future version of numpy, it will be understood as
(type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\pisar\Anaconda3\lib\site-packages\tensorflow\python\framework\dty
pes.py:518: FutureWarning: Passing (type, 1) or '1type' as a synonym of ty
pe is deprecated; in a future version of numpy, it will be understood as
(type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\pisar\Anaconda3\lib\site-packages\tensorflow\python\framework\dty
pes.py:519: FutureWarning: Passing (type, 1) or '1type' as a synonym of ty
pe is deprecated; in a future version of numpy, it will be understood as
(type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\pisar\Anaconda3\lib\site-packages\tensorflow\python\framework\dty
pes.py:520: FutureWarning: Passing (type, 1) or '1type' as a synonym of ty
pe is deprecated; in a future version of numpy, it will be understood as
(type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\pisar\Anaconda3\lib\site-packages\tensorflow\python\framework\dty
pes.py:525: FutureWarning: Passing (type, 1) or '1type' as a synonym of ty
pe is deprecated; in a future version of numpy, it will be understood as
(type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
C:\Users\pisar\Anaconda3\lib\site-packages\tensorboard\compat\tensorflow_s
tub\dtypes.py:541: FutureWarning: Passing (type, 1) or '1type' as a synony
m of type is deprecated; in a future version of numpy, it will be understo
od as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\pisar\Anaconda3\lib\site-packages\tensorboard\compat\tensorflow_s
tub\dtypes.py:542: FutureWarning: Passing (type, 1) or '1type' as a synony
m of type is deprecated; in a future version of numpy, it will be understo
od as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\pisar\Anaconda3\lib\site-packages\tensorboard\compat\tensorflow_s
tub\dtypes.py:543: FutureWarning: Passing (type, 1) or '1type' as a synony
m of type is deprecated; in a future version of numpy, it will be understo
od as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\pisar\Anaconda3\lib\site-packages\tensorboard\compat\tensorflow_s
tub\dtypes.py:544: FutureWarning: Passing (type, 1) or '1type' as a synony
m of type is deprecated; in a future version of numpy, it will be understo
od as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\pisar\Anaconda3\lib\site-packages\tensorboard\compat\tensorflow_s
tub\dtypes.py:545: FutureWarning: Passing (type, 1) or '1type' as a synony
m of type is deprecated; in a future version of numpy, it will be understo
od as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\pisar\Anaconda3\lib\site-packages\tensorboard\compat\tensorflow_s
tub\dtypes.py:550: FutureWarning: Passing (type, 1) or '1type' as a synony
m of type is deprecated; in a future version of numpy, it will be understo
od as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

In [2]:

```
df = pd.read_csv('./train.csv')
```

In [3]:

```
pixels = df.columns.drop('label')
y_column = 'label'
```

In [4]:

```
df.head()
```

Out[4]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pi
0	1	0	0	0	0	0	0	0	0	0	...	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	

5 rows × 785 columns

In [5]:

```
train, validation = train_test_split(df, test_size=0.2)
```

In [6]:

```
x_train = train[pixels].values
y_train = train[y_column].values

x_val = validation[pixels].values
y_val = validation[y_column].values
```

In [7]:

```
y_train = y_train.reshape((y_train.shape[0], 1))
y_val = y_val.reshape((y_val.shape[0], 1))
```

In [8]:

```
print(x_train.shape, y_train.shape)
```

```
(33600, 784) (33600, 1)
```

In [9]:

```
from sklearn.metrics import accuracy_score
```

1 Нейросеть состоит из 3 скрытых слоев с дропаутом 0,25 и оптимизатором RMSprop

In [10]:

```
y_all_pred = np.zeros((3, x_val.shape[0])).astype(np.int64) #matrix results
```

In [11]:

```
model1 = Sequential()
model1.add(Dense(units = 64, activation = 'relu', input_dim = len(pixels)))
model1.add(Dense(units = 32, activation = 'relu'))
model1.add(Dropout(0.25))
model1.add(Dense(units = 10, activation = 'softmax'))
```

```
WARNING:tensorflow:From C:\Users\pisar\Anaconda3\lib\site-packages\keras\b
ackend\tensorflow_backend.py:74: The name tf.get_default_graph is deprecat
ed. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\pisar\Anaconda3\lib\site-packages\keras\b
ackend\tensorflow_backend.py:517: The name tf.placeholder is deprecated. P
lease use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From C:\Users\pisar\Anaconda3\lib\site-packages\keras\b
ackend\tensorflow_backend.py:4138: The name tf.random_uniform is deprecate
d. Please use tf.random.uniform instead.

WARNING:tensorflow:From C:\Users\pisar\Anaconda3\lib\site-packages\keras\b
ackend\tensorflow_backend.py:133: The name tf.placeholder_with_default is
deprecated. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From C:\Users\pisar\Anaconda3\lib\site-packages\keras\b
ackend\tensorflow_backend.py:3445: calling dropout (from tensorflow.pytho
n.ops.nn_ops) with keep_prob is deprecated and will be removed in a future
version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1
- keep_prob`.
```

In [12]:

```
optimizer = RMSprop (lr = 0.001) #сколпрость обучения
```

In [13]:

```
model1.compile(optimizer = optimizer, loss = "sparse_categorical_crossentropy", metrics
=["accuracy"])
```

```
WARNING:tensorflow:From C:\Users\pisar\Anaconda3\lib\site-packages\keras\o
ptimizers.py:790: The name tf.train.Optimizer is deprecated. Please use t
f.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From C:\Users\pisar\Anaconda3\lib\site-packages\keras\b
ackend\tensorflow_backend.py:3341: The name tf.log is deprecated. Please u
se tf.math.log instead.
```

In [14]:

```
history1 = model1.fit(x_train, y_train, validation_data = (x_val, y_val), epochs = 50,
batch_size = 32)
```

```
WARNING:tensorflow:From C:\Users\pisar\Anaconda3\lib\site-packages\tensorf
low\python\ops\math_grad.py:1250: add_dispatch_support.<locals>.wrapper (f
rom tensorflow.python.ops.array_ops) is deprecated and will be removed in
a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 33600 samples, validate on 8400 samples
Epoch 1/50
33600/33600 [==============================] - 6s 172us/step - loss: 7.382
5 - acc: 0.5306 - val_loss: 5.5237 - val_acc: 0.6519
Epoch 2/50
33600/33600 [==============================] - 5s 162us/step - loss: 5.300
4 - acc: 0.6619 - val_loss: 3.6996 - val_acc: 0.7631
Epoch 3/50
33600/33600 [==============================] - 5s 155us/step - loss: 4.054
1 - acc: 0.7397 - val_loss: 3.0883 - val_acc: 0.8020
Epoch 4/50
33600/33600 [==============================] - 4s 129us/step - loss: 3.634
4 - acc: 0.7661 - val_loss: 3.0858 - val_acc: 0.8045
Epoch 5/50
33600/33600 [==============================] - 4s 132us/step - loss: 3.432
8 - acc: 0.7800 - val_loss: 3.2307 - val_acc: 0.7960
Epoch 6/50
33600/33600 [==============================] - 4s 122us/step - loss: 3.114
8 - acc: 0.7988 - val_loss: 1.9329 - val_acc: 0.8742
Epoch 7/50
33600/33600 [==============================] - 4s 117us/step - loss: 2.308
1 - acc: 0.8474 - val_loss: 1.6840 - val_acc: 0.8902
Epoch 8/50
33600/33600 [==============================] - 4s 116us/step - loss: 1.943
2 - acc: 0.8717 - val_loss: 1.3778 - val_acc: 0.9106
Epoch 9/50
33600/33600 [==============================] - 5s 161us/step - loss: 1.764
6 - acc: 0.8834 - val_loss: 1.3458 - val_acc: 0.9117
Epoch 10/50
33600/33600 [==============================] - 5s 138us/step - loss: 1.764
5 - acc: 0.8810 - val_loss: 1.3206 - val_acc: 0.9138
Epoch 11/50
33600/33600 [==============================] - 4s 132us/step - loss: 1.584
1 - acc: 0.8924 - val_loss: 1.1447 - val_acc: 0.9246
Epoch 12/50
33600/33600 [==============================] - 5s 138us/step - loss: 1.434
9 - acc: 0.8940 - val_loss: 1.1054 - val_acc: 0.9202
Epoch 13/50
33600/33600 [==============================] - 4s 132us/step - loss: 1.084
1 - acc: 0.8624 - val_loss: 0.9188 - val_acc: 0.8980
Epoch 14/50
33600/33600 [==============================] - 5s 145us/step - loss: 0.950
2 - acc: 0.8484 - val_loss: 0.8034 - val_acc: 0.8998
Epoch 15/50
33600/33600 [==============================] - 5s 138us/step - loss: 0.922
3 - acc: 0.8559 - val_loss: 0.8191 - val_acc: 0.8948
Epoch 16/50
33600/33600 [==============================] - 4s 130us/step - loss: 0.894
7 - acc: 0.8498 - val_loss: 0.8485 - val_acc: 0.8717
Epoch 17/50
33600/33600 [==============================] - 5s 152us/step - loss: 0.927
0 - acc: 0.8531 - val_loss: 0.8796 - val_acc: 0.9080
Epoch 18/50
33600/33600 [==============================] - 5s 135us/step - loss: 0.972
2 - acc: 0.8573 - val_loss: 0.9210 - val_acc: 0.8704
```

```
Epoch 19/50
33600/33600 [==============================] - 4s 132us/step - loss: 0.965
4 - acc: 0.8574 - val_loss: 0.7963 - val_acc: 0.8990
Epoch 20/50
33600/33600 [==============================] - 5s 142us/step - loss: 0.915
1 - acc: 0.8525 - val_loss: 0.9212 - val_acc: 0.8992
Epoch 21/50
33600/33600 [==============================] - 8s 229us/step - loss: 0.925
7 - acc: 0.8524 - val_loss: 0.9204 - val_acc: 0.8890
Epoch 22/50
33600/33600 [==============================] - 7s 222us/step - loss: 0.972
4 - acc: 0.8593 - val_loss: 0.8043 - val_acc: 0.9096
Epoch 23/50
33600/33600 [==============================] - 6s 169us/step - loss: 1.050
2 - acc: 0.8684 - val_loss: 0.8404 - val_acc: 0.9113
Epoch 24/50
33600/33600 [==============================] - 5s 163us/step - loss: 1.051
8 - acc: 0.8658 - val_loss: 1.2852 - val_acc: 0.8923
Epoch 25/50
33600/33600 [==============================] - 5s 141us/step - loss: 1.032
5 - acc: 0.8764 - val_loss: 0.8743 - val_acc: 0.9060
Epoch 26/50
33600/33600 [==============================] - 9s 266us/step - loss: 1.052
3 - acc: 0.8684 - val_loss: 0.8324 - val_acc: 0.8998
Epoch 27/50
33600/33600 [==============================] - 7s 209us/step - loss: 1.115
6 - acc: 0.8505 - val_loss: 0.7710 - val_acc: 0.8835
Epoch 28/50
33600/33600 [==============================] - 4s 123us/step - loss: 1.100
6 - acc: 0.8644 - val_loss: 0.8673 - val_acc: 0.9063
Epoch 29/50
33600/33600 [==============================] - 4s 126us/step - loss: 1.189
6 - acc: 0.8679 - val_loss: 0.9794 - val_acc: 0.9119
Epoch 30/50
33600/33600 [==============================] - 4s 132us/step - loss: 1.147
0 - acc: 0.8749 - val_loss: 0.9670 - val_acc: 0.8799
Epoch 31/50
33600/33600 [==============================] - 4s 122us/step - loss: 1.105
5 - acc: 0.8691 - val_loss: 1.1889 - val_acc: 0.9114
Epoch 32/50
33600/33600 [==============================] - 5s 147us/step - loss: 1.180
2 - acc: 0.8774 - val_loss: 0.8671 - val_acc: 0.9105
Epoch 33/50
33600/33600 [==============================] - 5s 153us/step - loss: 1.081
2 - acc: 0.8707 - val_loss: 1.0583 - val_acc: 0.8892
Epoch 34/50
33600/33600 [==============================] - 4s 119us/step - loss: 1.115
7 - acc: 0.8707 - val_loss: 0.9606 - val_acc: 0.8649
Epoch 35/50
33600/33600 [==============================] - 4s 123us/step - loss: 1.209
6 - acc: 0.8735 - val_loss: 0.9412 - val_acc: 0.9058
Epoch 36/50
33600/33600 [==============================] - 4s 125us/step - loss: 1.136
9 - acc: 0.8605 - val_loss: 1.0130 - val_acc: 0.9098
Epoch 37/50
33600/33600 [==============================] - 4s 114us/step - loss: 1.096
8 - acc: 0.8699 - val_loss: 1.0303 - val_acc: 0.8914
Epoch 38/50
33600/33600 [==============================] - 4s 124us/step - loss: 1.114
9 - acc: 0.8745 - val_loss: 0.9187 - val_acc: 0.9058
Epoch 39/50
```

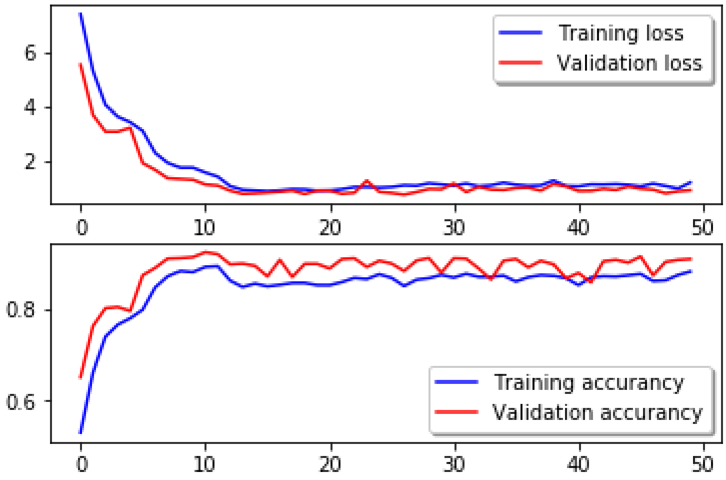
```
33600/33600 [==============================] - 4s 113us/step - loss: 1.285
8 - acc: 0.8730 - val_loss: 1.1578 - val_acc: 0.8983
Epoch 40/50
33600/33600 [==============================] - 4s 109us/step - loss: 1.090
1 - acc: 0.8678 - val_loss: 1.0756 - val_acc: 0.8675
Epoch 41/50
33600/33600 [==============================] - 4s 120us/step - loss: 1.077
4 - acc: 0.8525 - val_loss: 0.9058 - val_acc: 0.8792
Epoch 42/50
33600/33600 [==============================] - 4s 130us/step - loss: 1.149
4 - acc: 0.8695 - val_loss: 0.9120 - val_acc: 0.8585
Epoch 43/50
33600/33600 [==============================] - 4s 127us/step - loss: 1.143
1 - acc: 0.8721 - val_loss: 0.9847 - val_acc: 0.9046
Epoch 44/50
33600/33600 [==============================] - 4s 127us/step - loss: 1.160
1 - acc: 0.8713 - val_loss: 0.9424 - val_acc: 0.9080
Epoch 45/50
33600/33600 [==============================] - 4s 122us/step - loss: 1.132
2 - acc: 0.8741 - val_loss: 1.0697 - val_acc: 0.9019
Epoch 46/50
33600/33600 [==============================] - 7s 199us/step - loss: 1.075
0 - acc: 0.8771 - val_loss: 0.9915 - val_acc: 0.9154
Epoch 47/50
33600/33600 [==============================] - 4s 131us/step - loss: 1.179
4 - acc: 0.8616 - val_loss: 0.9551 - val_acc: 0.8740
Epoch 48/50
33600/33600 [==============================] - 4s 117us/step - loss: 1.091
9 - acc: 0.8631 - val_loss: 0.8221 - val_acc: 0.9033
Epoch 49/50
33600/33600 [==============================] - 4s 113us/step - loss: 0.996
2 - acc: 0.8746 - val_loss: 0.8941 - val_acc: 0.9079
Epoch 50/50
33600/33600 [==============================] - 4s 113us/step - loss: 1.216
4 - acc: 0.8824 - val_loss: 0.9334 - val_acc: 0.9096
```

In [15]:

```python
def plotgraph(history):
    fig, ax = plt.subplots(2,1)
    ax[0].plot(history.history['loss'], color = 'b', label = "Training loss")
    ax[0].plot(history.history['val_loss'], color = 'r', label = "Validation loss", axe
s = ax[0])
    legend = ax[0].legend(loc = 'best', shadow = True)
    ax[1].plot(history.history['acc'], color = 'b', label = "Training accurancy")
    ax[1].plot(history.history['val_acc'], color = 'r', label = "Validation accurancy")
    legend = ax[1].legend(loc = 'best', shadow = True)
```

In [16]:

```
plotgraph(history1)
```



In [17]:

```
y_pred_mod1 = model1.predict_classes(x_val)
accuracy_score(y_val, y_pred_mod1)
```

Out[17]:

```
0.9096428571428572
```

In [18]:

```
y_all_pred[0] = y_pred_mod1
```

2 Нейросеть с 4 скрытыми слоями дропаутом 0,25 и оптимизатором Adagrad

In [19]:

```
model2 = Sequential()
model2.add(Dense(256, activation = 'relu', input_dim = len(pixels)))
model2.add(Dense(128, activation = 'relu'))
model2.add(Dense(64, activation = 'relu'))
model2.add(Dropout(0.25))
model2.add(Dense(10, activation = 'softmax'))
```

In [20]:

```
optimizer2 = Adagrad()
```

In [21]:

```
model2.compile(optimizer = optimizer2, loss = "sparse_categorical_crossentropy", metrics=["accuracy"])
```

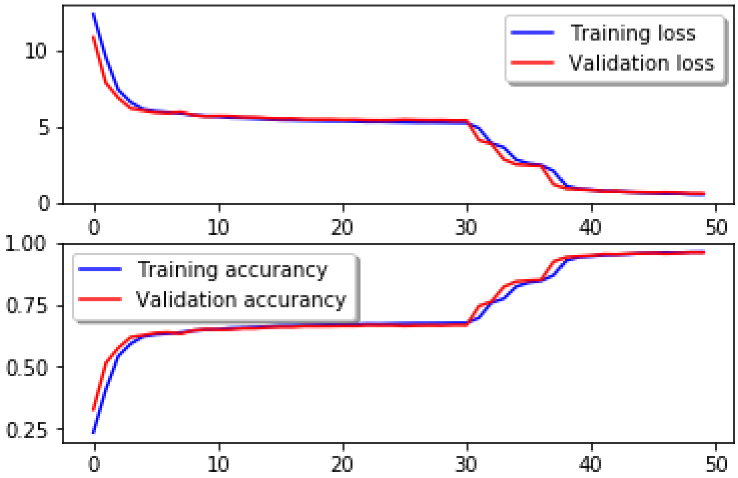
In [22]:

```
history2 = model2.fit(x_train, y_train, validation_data = (x_val, y_val), epochs = 50,
batch_size = 32)
plotgraph(history2)
```

```
Train on 33600 samples, validate on 8400 samples
Epoch 1/50
33600/33600 [==============================] - 9s 253us/step - loss: 12.33
73 - acc: 0.2333 - val_loss: 10.8237 - val_acc: 0.3264
Epoch 2/50
33600/33600 [==============================] - 9s 255us/step - loss: 9.525
9 - acc: 0.4073 - val_loss: 7.8256 - val_acc: 0.5126
Epoch 3/50
33600/33600 [==============================] - 8s 238us/step - loss: 7.369
1 - acc: 0.5400 - val_loss: 6.8460 - val_acc: 0.5731
Epoch 4/50
33600/33600 [==============================] - 8s 224us/step - loss: 6.569
1 - acc: 0.5903 - val_loss: 6.1601 - val_acc: 0.6168
Epoch 5/50
33600/33600 [==============================] - 7s 221us/step - loss: 6.103
0 - acc: 0.6194 - val_loss: 6.0230 - val_acc: 0.6248
Epoch 6/50
33600/33600 [==============================] - 8s 223us/step - loss: 5.973
2 - acc: 0.6279 - val_loss: 5.8853 - val_acc: 0.6340
Epoch 7/50
33600/33600 [==============================] - 8s 229us/step - loss: 5.906
4 - acc: 0.6318 - val_loss: 5.8470 - val_acc: 0.6367
Epoch 8/50
33600/33600 [==============================] - 10s 291us/step - loss: 5.83
93 - acc: 0.6361 - val_loss: 5.9245 - val_acc: 0.6311
Epoch 9/50
33600/33600 [==============================] - 8s 233us/step - loss: 5.723
6 - acc: 0.6434 - val_loss: 5.7140 - val_acc: 0.6444
Epoch 10/50
33600/33600 [==============================] - 8s 241us/step - loss: 5.658
4 - acc: 0.6476 - val_loss: 5.6351 - val_acc: 0.6493
Epoch 11/50
33600/33600 [==============================] - 8s 245us/step - loss: 5.629
5 - acc: 0.6491 - val_loss: 5.6664 - val_acc: 0.6476
Epoch 12/50
33600/33600 [==============================] - 8s 244us/step - loss: 5.555
3 - acc: 0.6537 - val_loss: 5.6511 - val_acc: 0.6482
Epoch 13/50
33600/33600 [==============================] - 8s 250us/step - loss: 5.536
5 - acc: 0.6552 - val_loss: 5.5866 - val_acc: 0.6525
Epoch 14/50
33600/33600 [==============================] - 8s 248us/step - loss: 5.499
5 - acc: 0.6574 - val_loss: 5.5825 - val_acc: 0.6526
Epoch 15/50
33600/33600 [==============================] - 8s 245us/step - loss: 5.460
2 - acc: 0.6601 - val_loss: 5.5185 - val_acc: 0.6567
Epoch 16/50
33600/33600 [==============================] - 8s 250us/step - loss: 5.412
7 - acc: 0.6632 - val_loss: 5.4896 - val_acc: 0.6581
Epoch 17/50
33600/33600 [==============================] - 8s 249us/step - loss: 5.397
4 - acc: 0.6637 - val_loss: 5.4942 - val_acc: 0.6581
Epoch 18/50
33600/33600 [==============================] - 8s 252us/step - loss: 5.376
8 - acc: 0.6650 - val_loss: 5.4445 - val_acc: 0.6615
Epoch 19/50
33600/33600 [==============================] - 9s 254us/step - loss: 5.360
8 - acc: 0.6662 - val_loss: 5.4453 - val_acc: 0.6613
Epoch 20/50
33600/33600 [==============================] - 9s 256us/step - loss: 5.344
6 - acc: 0.6674 - val_loss: 5.4459 - val_acc: 0.6617
```

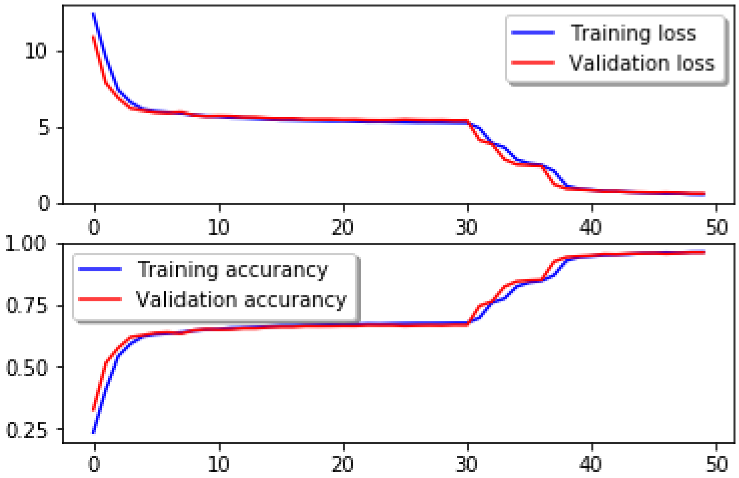
```
Epoch 21/50
33600/33600 [==============================] - 9s 255us/step - loss: 5.347
1 - acc: 0.6671 - val_loss: 5.4200 - val_acc: 0.6627
Epoch 22/50
33600/33600 [==============================] - 8s 251us/step - loss: 5.322
1 - acc: 0.6686 - val_loss: 5.4248 - val_acc: 0.6625
Epoch 23/50
33600/33600 [==============================] - 9s 254us/step - loss: 5.287
8 - acc: 0.6709 - val_loss: 5.3906 - val_acc: 0.6650
Epoch 24/50
33600/33600 [==============================] - 8s 251us/step - loss: 5.292
7 - acc: 0.6703 - val_loss: 5.3826 - val_acc: 0.6646
Epoch 25/50
33600/33600 [==============================] - 9s 256us/step - loss: 5.272
6 - acc: 0.6717 - val_loss: 5.3917 - val_acc: 0.6645
Epoch 26/50
33600/33600 [==============================] - 9s 264us/step - loss: 5.261
2 - acc: 0.6727 - val_loss: 5.4297 - val_acc: 0.6625
Epoch 27/50
33600/33600 [==============================] - 9s 260us/step - loss: 5.245
7 - acc: 0.6733 - val_loss: 5.4029 - val_acc: 0.6639
Epoch 28/50
33600/33600 [==============================] - 9s 256us/step - loss: 5.244
8 - acc: 0.6734 - val_loss: 5.3884 - val_acc: 0.6646
Epoch 29/50
33600/33600 [==============================] - 9s 255us/step - loss: 5.237
2 - acc: 0.6738 - val_loss: 5.3959 - val_acc: 0.6640
Epoch 30/50
33600/33600 [==============================] - 9s 257us/step - loss: 5.225
3 - acc: 0.6746 - val_loss: 5.3678 - val_acc: 0.6660
Epoch 31/50
33600/33600 [==============================] - 9s 278us/step - loss: 5.217
7 - acc: 0.6749 - val_loss: 5.3797 - val_acc: 0.6655
Epoch 32/50
33600/33600 [==============================] - 9s 261us/step - loss: 4.868
5 - acc: 0.6952 - val_loss: 4.0802 - val_acc: 0.7430
Epoch 33/50
33600/33600 [==============================] - 10s 287us/step - loss: 3.88
16 - acc: 0.7565 - val_loss: 3.8283 - val_acc: 0.7611
Epoch 34/50
33600/33600 [==============================] - 12s 344us/step - loss: 3.60
25 - acc: 0.7728 - val_loss: 2.8157 - val_acc: 0.8206
Epoch 35/50
33600/33600 [==============================] - 13s 382us/step - loss: 2.79
21 - acc: 0.8217 - val_loss: 2.4785 - val_acc: 0.8421
Epoch 36/50
33600/33600 [==============================] - 9s 275us/step - loss: 2.545
4 - acc: 0.8376 - val_loss: 2.4254 - val_acc: 0.8462
Epoch 37/50
33600/33600 [==============================] - 9s 257us/step - loss: 2.439
1 - acc: 0.8445 - val_loss: 2.3795 - val_acc: 0.8490
Epoch 38/50
33600/33600 [==============================] - 9s 264us/step - loss: 2.055
6 - acc: 0.8673 - val_loss: 1.1665 - val_acc: 0.9221
Epoch 39/50
33600/33600 [==============================] - 9s 275us/step - loss: 1.060
6 - acc: 0.9270 - val_loss: 0.8714 - val_acc: 0.9405
Epoch 40/50
33600/33600 [==============================] - 10s 292us/step - loss: 0.86
42 - acc: 0.9404 - val_loss: 0.8354 - val_acc: 0.9439
Epoch 41/50
```

```
33600/33600 [==============================] - 10s 297us/step - loss: 0.80
23 - acc: 0.9438 - val_loss: 0.7818 - val_acc: 0.9468
Epoch 42/50
33600/33600 [==============================] - 8s 251us/step - loss: 0.727
4 - acc: 0.9485 - val_loss: 0.7000 - val_acc: 0.9524
Epoch 43/50
33600/33600 [==============================] - 10s 295us/step - loss: 0.70
40 - acc: 0.9501 - val_loss: 0.7170 - val_acc: 0.9513
Epoch 44/50
33600/33600 [==============================] - 21s 626us/step - loss: 0.66
88 - acc: 0.9522 - val_loss: 0.6548 - val_acc: 0.9545
Epoch 45/50
33600/33600 [==============================] - 11s 335us/step - loss: 0.61
80 - acc: 0.9564 - val_loss: 0.6481 - val_acc: 0.9554
Epoch 46/50
33600/33600 [==============================] - 13s 384us/step - loss: 0.59
14 - acc: 0.9574 - val_loss: 0.6238 - val_acc: 0.9565
Epoch 47/50
33600/33600 [==============================] - 10s 294us/step - loss: 0.56
03 - acc: 0.9594 - val_loss: 0.6409 - val_acc: 0.9539
Epoch 48/50
33600/33600 [==============================] - 11s 323us/step - loss: 0.57
12 - acc: 0.9581 - val_loss: 0.6148 - val_acc: 0.9563
Epoch 49/50
33600/33600 [==============================] - 10s 310us/step - loss: 0.51
79 - acc: 0.9607 - val_loss: 0.5719 - val_acc: 0.9593
Epoch 50/50
33600/33600 [==============================] - 10s 297us/step - loss: 0.50
54 - acc: 0.9614 - val_loss: 0.5733 - val_acc: 0.9588
```



In [24]:

```
plotgraph(history2)
```



In [25]:

```
y_pred_mod2 = model2.predict_classes(x_val)
accuracy_score(y_val, y_pred_mod2)
```

Out[25]:

```
0.9588095238095238
```

In [26]:

```
y_all_pred[1] = y_pred_mod2
```

Применим к этой модели другой оптимизатор - Adadelta

In [30]:

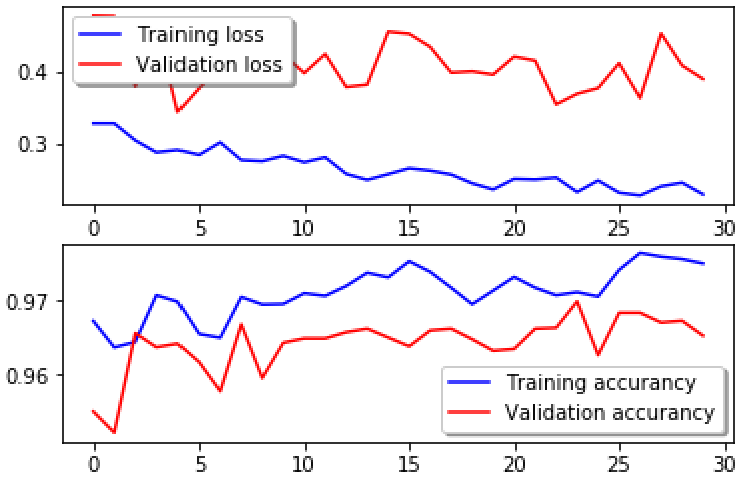
```
optimizer3 = Adadelta()
model2.compile(optimizer = optimizer3, loss = "sparse_categorical_crossentropy", metric
s=["accuracy"])
```

In [31]:

```
history3 = model2.fit(x_train, y_train, validation_data = (x_val, y_val), epochs = 30,
batch_size = 64)
plotgraph(history3)
```

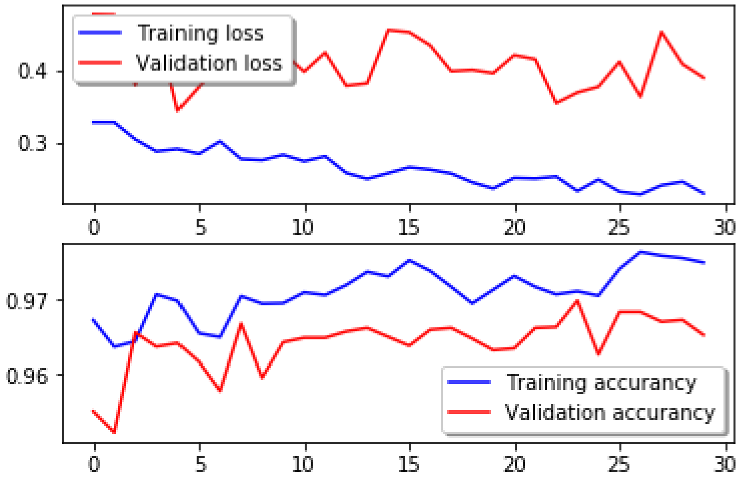
```
Train on 33600 samples, validate on 8400 samples
Epoch 1/30
33600/33600 [==============================] - 6s 191us/step - loss: 0.327
9 - acc: 0.9672 - val_loss: 0.4772 - val_acc: 0.9550
Epoch 2/30
33600/33600 [==============================] - 6s 171us/step - loss: 0.327
8 - acc: 0.9637 - val_loss: 0.4766 - val_acc: 0.9521
Epoch 3/30
33600/33600 [==============================] - 6s 187us/step - loss: 0.304
3 - acc: 0.9644 - val_loss: 0.3788 - val_acc: 0.9656
Epoch 4/30
33600/33600 [==============================] - 6s 168us/step - loss: 0.287
7 - acc: 0.9707 - val_loss: 0.4642 - val_acc: 0.9637
Epoch 5/30
33600/33600 [==============================] - 5s 148us/step - loss: 0.291
0 - acc: 0.9699 - val_loss: 0.3439 - val_acc: 0.9642
Epoch 6/30
33600/33600 [==============================] - 5s 148us/step - loss: 0.284
2 - acc: 0.9655 - val_loss: 0.3766 - val_acc: 0.9617
Epoch 7/30
33600/33600 [==============================] - 5s 148us/step - loss: 0.301
5 - acc: 0.9650 - val_loss: 0.4072 - val_acc: 0.9577
Epoch 8/30
33600/33600 [==============================] - 5s 151us/step - loss: 0.277
1 - acc: 0.9705 - val_loss: 0.4151 - val_acc: 0.9668
Epoch 9/30
33600/33600 [==============================] - 5s 151us/step - loss: 0.275
5 - acc: 0.9695 - val_loss: 0.4269 - val_acc: 0.9595
Epoch 10/30
33600/33600 [==============================] - 5s 155us/step - loss: 0.282
9 - acc: 0.9695 - val_loss: 0.4237 - val_acc: 0.9643
Epoch 11/30
33600/33600 [==============================] - 5s 153us/step - loss: 0.274
0 - acc: 0.9710 - val_loss: 0.3977 - val_acc: 0.9649
Epoch 12/30
33600/33600 [==============================] - 5s 150us/step - loss: 0.280
9 - acc: 0.9706 - val_loss: 0.4241 - val_acc: 0.9649
Epoch 13/30
33600/33600 [==============================] - 5s 161us/step - loss: 0.257
8 - acc: 0.9720 - val_loss: 0.3783 - val_acc: 0.9657
Epoch 14/30
33600/33600 [==============================] - 5s 162us/step - loss: 0.249
5 - acc: 0.9738 - val_loss: 0.3816 - val_acc: 0.9662
Epoch 15/30
33600/33600 [==============================] - 5s 148us/step - loss: 0.257
5 - acc: 0.9731 - val_loss: 0.4551 - val_acc: 0.9650
Epoch 16/30
33600/33600 [==============================] - 5s 146us/step - loss: 0.265
6 - acc: 0.9753 - val_loss: 0.4521 - val_acc: 0.9638
Epoch 17/30
33600/33600 [==============================] - 5s 149us/step - loss: 0.262
3 - acc: 0.9739 - val_loss: 0.4339 - val_acc: 0.9660
Epoch 18/30
33600/33600 [==============================] - 5s 148us/step - loss: 0.257
0 - acc: 0.9717 - val_loss: 0.3983 - val_acc: 0.9662
Epoch 19/30
33600/33600 [==============================] - 5s 153us/step - loss: 0.244
8 - acc: 0.9695 - val_loss: 0.4000 - val_acc: 0.9648
Epoch 20/30
33600/33600 [==============================] - 5s 148us/step - loss: 0.236
2 - acc: 0.9713 - val_loss: 0.3956 - val_acc: 0.9632
```

```
Epoch 21/30
33600/33600 [==============================] - 6s 169us/step - loss: 0.250
9 - acc: 0.9732 - val_loss: 0.4205 - val_acc: 0.9635
Epoch 22/30
33600/33600 [==============================] - 5s 151us/step - loss: 0.250
0 - acc: 0.9717 - val_loss: 0.4149 - val_acc: 0.9662
Epoch 23/30
33600/33600 [==============================] - 5s 151us/step - loss: 0.252
7 - acc: 0.9707 - val_loss: 0.3543 - val_acc: 0.9663
Epoch 24/30
33600/33600 [==============================] - 5s 162us/step - loss: 0.232
7 - acc: 0.9711 - val_loss: 0.3689 - val_acc: 0.9699
Epoch 25/30
33600/33600 [==============================] - 5s 160us/step - loss: 0.248
7 - acc: 0.9705 - val_loss: 0.3768 - val_acc: 0.9626
Epoch 26/30
33600/33600 [==============================] - 5s 150us/step - loss: 0.231
8 - acc: 0.9741 - val_loss: 0.4114 - val_acc: 0.9683
Epoch 27/30
33600/33600 [==============================] - 5s 145us/step - loss: 0.228
0 - acc: 0.9764 - val_loss: 0.3630 - val_acc: 0.9683
Epoch 28/30
33600/33600 [==============================] - 5s 155us/step - loss: 0.240
6 - acc: 0.9759 - val_loss: 0.4526 - val_acc: 0.9670
Epoch 29/30
33600/33600 [==============================] - 5s 159us/step - loss: 0.245
7 - acc: 0.9756 - val_loss: 0.4082 - val_acc: 0.9673
Epoch 30/30
33600/33600 [==============================] - 5s 153us/step - loss: 0.229
5 - acc: 0.9750 - val_loss: 0.3894 - val_acc: 0.9652
```



In [32]:

```
plotgraph(history3)
```



In [33]:

```
y_pred_mod3 = model2.predict_classes(x_val)
accuracy_score(y_val, y_pred_mod3)
```

Out[33]:

```
0.9652380952380952
```

In [34]:

```
y_all_pred[2] = y_pred_mod3
```

Как видим две простые полносвязные нейросети показали довольно посредственные результаты, лучший из которых 0.95

3 Нейросеть -сверточная, должна показать лучший результат

In [35]:

```
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Reshape, Conv2D, AveragePooling2D, Flatten
from keras.layers import MaxPooling2D
from keras.optimizers import adam

y_train_cnn = keras.utils.to_categorical(y_train, num_classes=10) #преобразует метки классов в бинарную матрицу
```

In [37]:

```
model_cnn = Sequential()
model_cnn.add(Reshape(target_shape=(1, 28, 28), input_shape=(784,)))
model_cnn.add(Conv2D(kernel_size=(3, 3), filters=6, padding="same", data_format="channels_first", kernel_initializer="uniform", use_bias=False))
model_cnn.add(MaxPooling2D(pool_size=(2, 2), data_format="channels_first"))
model_cnn.add(Conv2D(kernel_size=(5, 5), filters=16, padding="same", data_format="channels_first", kernel_initializer="uniform", use_bias=False))
model_cnn.add(MaxPooling2D(pool_size=(2, 2), data_format="channels_first"))
model_cnn.add(Conv2D(kernel_size=(5, 5), filters=120, padding="same", data_format="channels_first", kernel_initializer="uniform", use_bias=False))
model_cnn.add(Flatten())
model_cnn.add(Dense(output_dim=120, activation='relu'))
model_cnn.add(Dropout(0.25))
model_cnn.add(Dense(output_dim=120, activation='relu'))
model_cnn.add(Dense(output_dim=10, activation='softmax'))
```

```
C:\Users\pisar\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: UserWa
rning: Update your `Dense` call to the Keras 2 API: `Dense(activation="rel
u", units=120)`
  if __name__ == '__main__':
C:\Users\pisar\Anaconda3\lib\site-packages\ipykernel_launcher.py:11: UserW
arning: Update your `Dense` call to the Keras 2 API: `Dense(activation="re
lu", units=120)`
  # This is added back by InteractiveShellApp.init_path()
C:\Users\pisar\Anaconda3\lib\site-packages\ipykernel_launcher.py:12: UserW
arning: Update your `Dense` call to the Keras 2 API: `Dense(activation="so
ftmax", units=10)`
  if sys.path[0] == '':
```

In [38]:

```
adam = keras.optimizers.Adam(lr=0.0005, beta_1=0.9, beta_2=0.999, epsilon=1e-08)
model_cnn.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
```

In [39]:

```
model_cnn.fit(x_train, y_train_cnn, epochs=30, batch_size=64)
```

```
Epoch 1/30
33600/33600 [==============================] - 33s 970us/step - loss: 0.26
42 - acc: 0.9179
Epoch 2/30
33600/33600 [==============================] - 29s 878us/step - loss: 0.10
93 - acc: 0.9668
Epoch 3/30
33600/33600 [==============================] - 29s 873us/step - loss: 0.08
37 - acc: 0.9736
Epoch 4/30
33600/33600 [==============================] - 32s 953us/step - loss: 0.07
26 - acc: 0.9774
Epoch 5/30
33600/33600 [==============================] - 28s 820us/step - loss: 0.06
57 - acc: 0.9796
Epoch 6/30
33600/33600 [==============================] - 29s 876us/step - loss: 0.05
99 - acc: 0.9816
Epoch 7/30
33600/33600 [==============================] - 32s 955us/step - loss: 0.06
00 - acc: 0.9821
Epoch 8/30
33600/33600 [==============================] - 29s 874us/step - loss: 0.05
19 - acc: 0.9845
Epoch 9/30
33600/33600 [==============================] - 29s 868us/step - loss: 0.05
99 - acc: 0.9827
Epoch 10/30
33600/33600 [==============================] - 30s 897us/step - loss: 0.05
47 - acc: 0.9843
Epoch 11/30
33600/33600 [==============================] - 31s 928us/step - loss: 0.04
45 - acc: 0.9863
Epoch 12/30
33600/33600 [==============================] - 34s 1ms/step - loss: 0.0490
- acc: 0.9858
Epoch 13/30
33600/33600 [==============================] - 32s 963us/step - loss: 0.04
86 - acc: 0.9855
Epoch 14/30
33600/33600 [==============================] - 27s 794us/step - loss: 0.04
20 - acc: 0.9876
Epoch 15/30
33600/33600 [==============================] - 30s 888us/step - loss: 0.05
19 - acc: 0.9855
Epoch 16/30
33600/33600 [==============================] - 31s 917us/step - loss: 0.04
04 - acc: 0.9882
Epoch 17/30
33600/33600 [==============================] - 31s 924us/step - loss: 0.05
04 - acc: 0.9860
Epoch 18/30
33600/33600 [==============================] - 28s 823us/step - loss: 0.02
92 - acc: 0.9913
Epoch 19/30
33600/33600 [==============================] - 29s 859us/step - loss: 0.04
22 - acc: 0.9876
Epoch 20/30
33600/33600 [==============================] - 31s 923us/step - loss: 0.04
51 - acc: 0.98740s - loss: 0.0450 - acc: 0.987
Epoch 21/30
```

```
33600/33600 [==============================] - 27s 793us/step - loss: 0.03
59 - acc: 0.9897
Epoch 22/30
33600/33600 [==============================] - 27s 795us/step - loss: 0.03
57 - acc: 0.9898
Epoch 23/30
33600/33600 [==============================] - 29s 876us/step - loss: 0.03
85 - acc: 0.9890
Epoch 24/30
33600/33600 [==============================] - 26s 788us/step - loss: 0.03
72 - acc: 0.9893
Epoch 25/30
33600/33600 [==============================] - 29s 867us/step - loss: 0.03
98 - acc: 0.9894
Epoch 26/30
33600/33600 [==============================] - 29s 849us/step - loss: 0.03
27 - acc: 0.9908
Epoch 27/30
33600/33600 [==============================] - 30s 897us/step - loss: 0.03
18 - acc: 0.9915
Epoch 28/30
33600/33600 [==============================] - 29s 864us/step - loss: 0.03
68 - acc: 0.9904
Epoch 29/30
33600/33600 [==============================] - 29s 869us/step - loss: 0.04
00 - acc: 0.98951s
Epoch 30/30
33600/33600 [==============================] - 29s 850us/step - loss: 0.03
17 - acc: 0.9915
```

Out[39]:

```
<keras.callbacks.History at 0x24ba14961d0>
```

In [40]:

```
y_pred_cnn = model_cnn.predict_classes(x_val)
```

In [41]:

```
accuracy_score(y_val, y_pred_cnn)
```

Out[41]:

```
0.9841666666666666
```

4 Нейросеть - ансамбль из слабых нейросетей

In [55]:

```
y_ensem_pred = np.zeros((x_val.shape[0],))
for i,line in enumerate(y_all_pred.T):
    y_ensem_pred[i] = np.argmax(np.bincount(line))
print(y_ensem_pred.shape, y_ensem_pred)
y_ensem_pred = y_ensem_pred.astype("int64")
```

```
(8400,) [4. 3. 8. ... 1. 4. 0.]
```

In [56]:

```
accuracy_score(y_ensem_pred, y_val)
```

Out[56]:

```
0.963452380952381
```

Качество улучшилось

In [ ]:

```

```