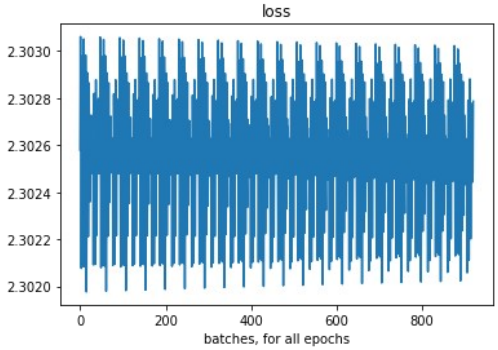
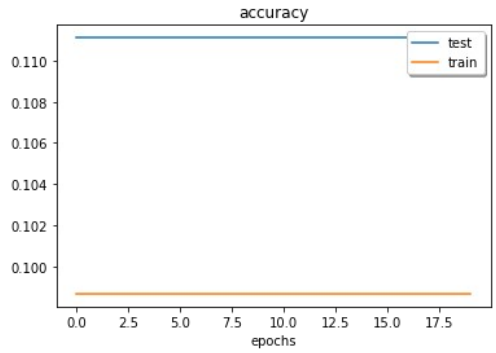
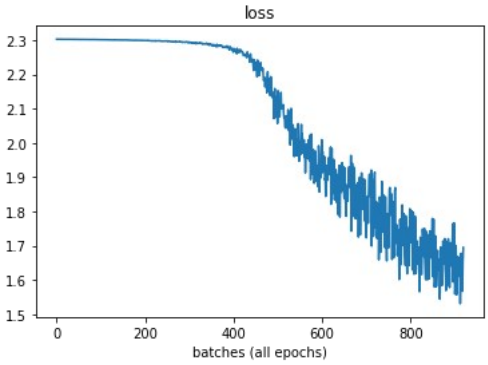
Ass02 vavoulis giorgos 1040156

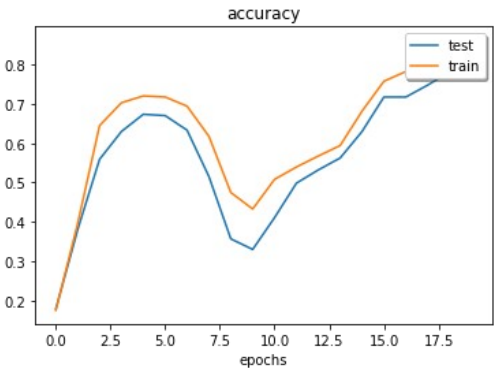
- **Default configuration** (**n_hid1**=15, **n_hid2**=0, **learning_rate**=0.01, **activation_function1**=sigmoid, **activation_function2**=none, **batch_size**=32, **optimizer**=SGD, **epochs=**20, **l2_regularization=**no)



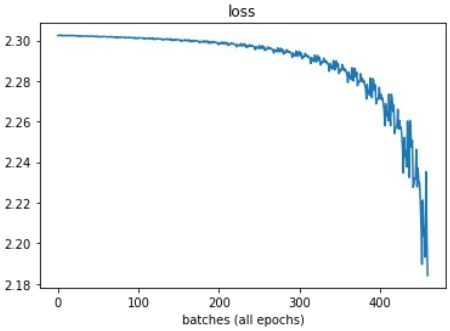


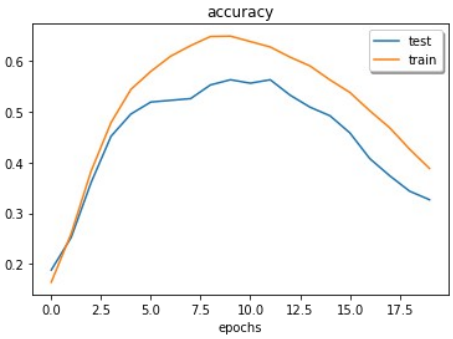
- (100, 0, 0.1, relu, none, 32, SGD, 20, no)



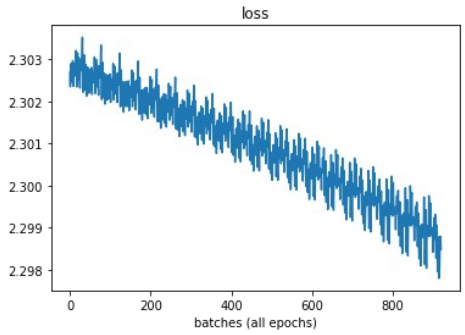


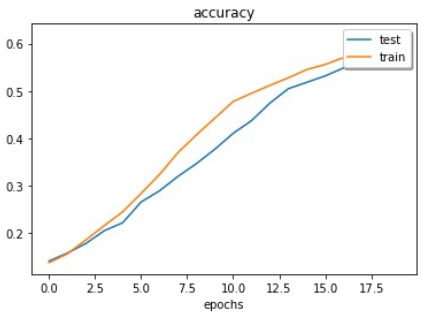
- (100, 0, 0.1, relu, none, 64, SGD,20, no)



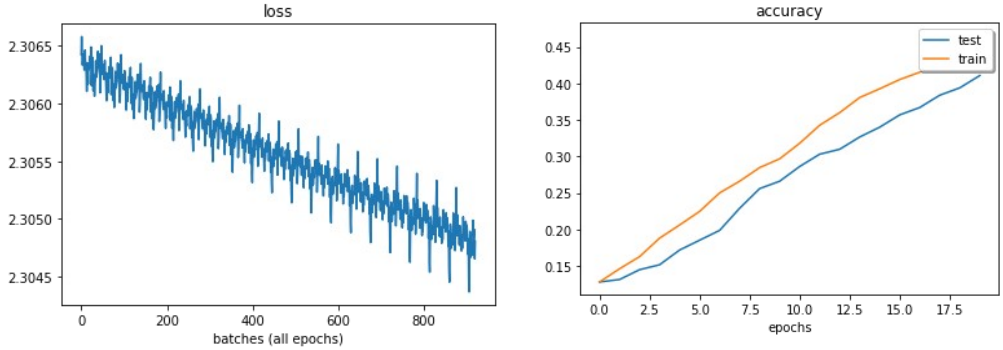


- (100, 0, 0.01, tanh, none, 32, SGD,20, no)

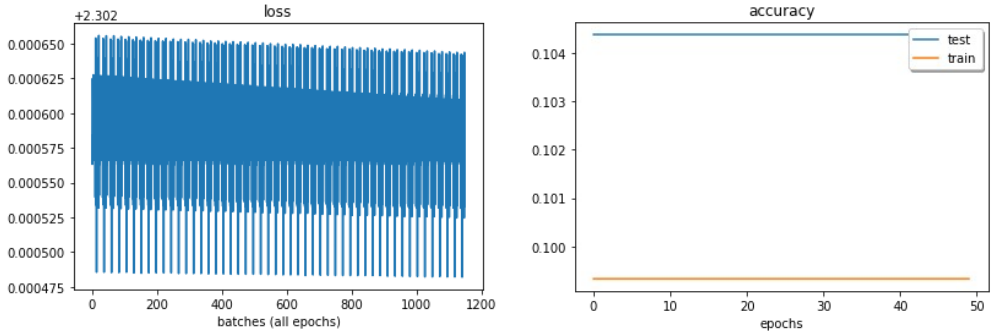




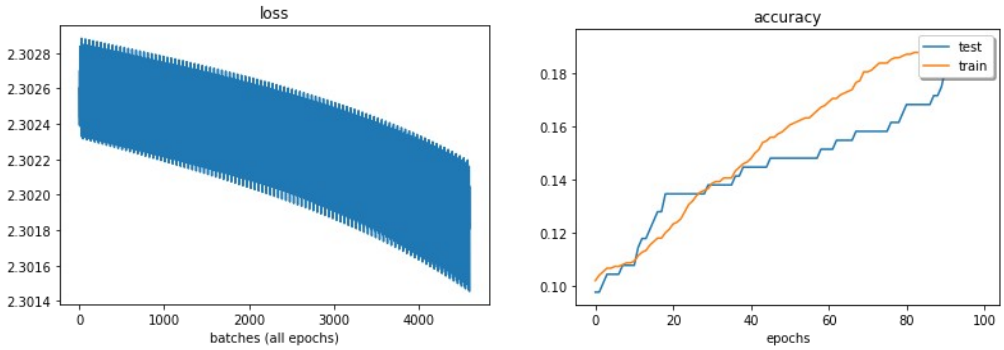
- (100, 0, 0.01, relu, none, 32, SGD, 20, yes)



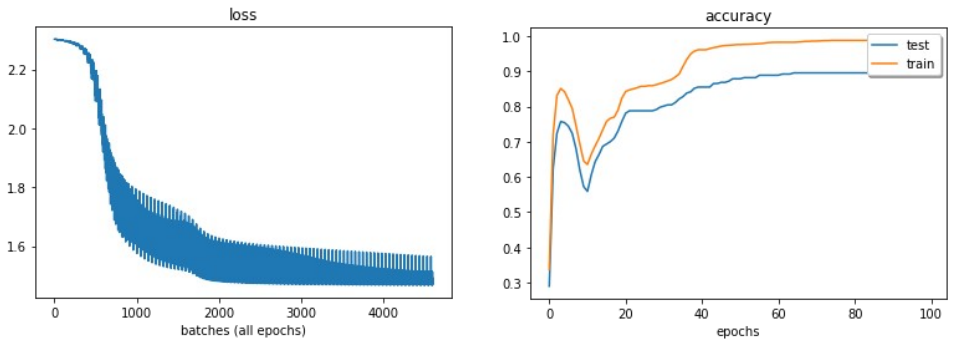
- (100, 100, 0.01, tanh, relu, 64, SGD, 50, yes)



- (200, 100, 0.01, tanh, tanh, 32, SGD, 100, yes)



- (100, 0, 0.05, tanh, none, 32, SGD, 100, yes) ← <u>BEST:</u> train accuracy: .989, test accuracy: .896

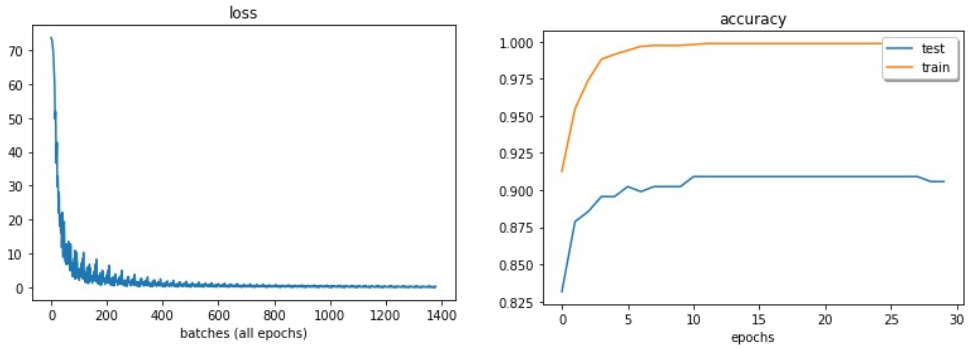


Finally I tried a manual definition of the loss function instead of the built-in and got better and more stable results. Maybe there is some bug in the built-in loss.

Manual: -np.sum(tf.reduce_sum(y * tf.log(out_layerout)))
Built-in: tf.nn.softmax_cross_entropy_with_logits(logits, labels).

So, last attempt with the manual loss function was:

- (50, 0, 0.01, sigmoid, none, 32, SGD, 30, yes) ← train accuracy: .999
  test accuracy: .906



So **best** train accuracy is **.999** and best test accuracy is **.906**.

Conclusions:

Sigmoid performance is disappointing, for the built-in loss function.
Adding more layers isn't always the best thing to do.