Assignment 3: Report basic

1) Checking the gradients for the generic K-NN

In order to check the correctness of the gradients I used the following formula

$$\frac{|g_a - g_n|}{\max(\text{eps}, |g_a| + |g_n|)} \quad \text{where eps a very small positive number}$$

where g_a is the analytically computed gradient and g_n is the numerically computed gradient and made sure that this relative error is small for each component of the gradient(bias gradient and weights gradient). To compute the numerical gradients I used the function ***ComputeGradsNumSlow.py.***

In order to check the gradients we decrease the dimension of the input to 50. lambda= 0.1 and all hidden layers have 20 nodes.

Gradients without batch normalization

2 Layers Network

```
LAYER: 1
Sum of relative weights error for Layer 1 : 0.00014673426882692818196
Sum of relative biases error for Layer 1 : -7.229957822585429938e-08

LAYER: 2
Sum of relative weights error for Layer 2 : 2.7203157320845678246e-06
Sum of relative biases error for Layer 2 : 2.1396981420099051635e-08
```

3 Layers Network

```
LAYER: 1
Sum of relative weights error for Layer 1 : -0.0018845415368999968391
Sum of relative biases error for Layer 1 : -1.1746269553369477418e-07

LAYER: 2
Sum of relative weights error for Layer 2 : 1.862968508467330108e-07
Sum of relative biases error for Layer 2 : 3.027183788983597292e-08

LAYER: 3
Sum of relative weights error for Layer 3 : 1.9484145334911440084e-06
Sum of relative biases error for Layer 3 : 1.0079336918441644 7875e-08
```

Gradients with batch normalization

2 Layers Network

```
LAYER: 1
Sum of relative weights error for Layer FC : 4.42125333445057e-05
Sum of relative betas error for Layer BN: : 8.646127578832108e-07
Sum of relative gammas error for Layer BN: : 9.415384832137367e-07

LAYER: 2
Sum of relative weights error for Layer FC : 8.013386982254017e-06
Sum of relative betas error for Layer BN: : 9.011532476209387e-07
Sum of relative gammas error for Layer BN: : 3.3187973562792394e-08
```

3 Layers Network

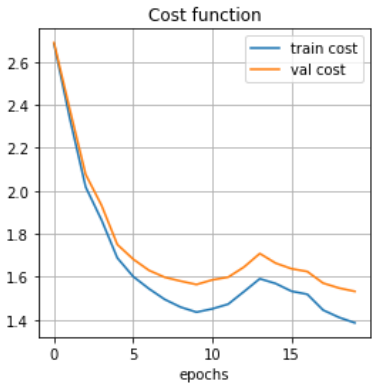
```
LAYER: 1
Sum of relative weights error for Layer FC : 6.552163728967155e-05
Sum of relative betas error for Layer BN: : 2.421152420043049e-06
Sum of relative gammas error for Layer BN: : 7.53856764889225e-06

LAYER: 2
Sum of relative weights error for Layer FC : 1.9291068433010034e-05
Sum of relative betas error for Layer BN: : 1.2539411495296493e-06
Sum of relative gammas error for Layer BN: : 1.1830048707082187e-06

LAYER: 3
Sum of relative weights error for Layer FC : 4.285723777639723e-06
Sum of relative betas error for Layer BN: : 3.3288229888493277e-06
Sum of relative gammas error for Layer BN: : 1.1156746088136048e-08
```

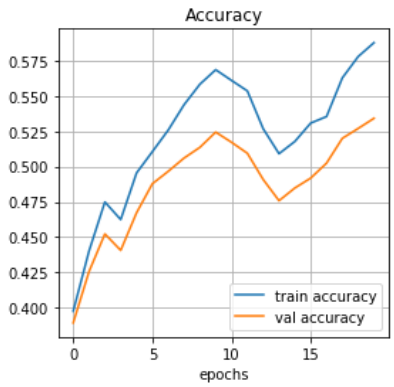
From the low relative error we can conclude that the gradients are implemented correctly.

2) 3-layer network with He initialization

- **3-layer NN without Batch Normalization**

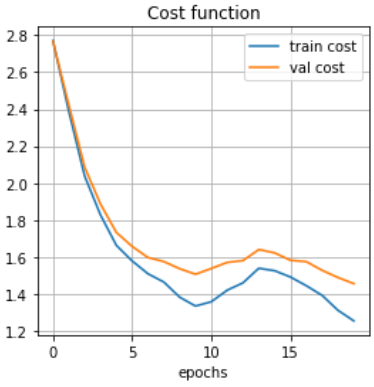
Train set accuracy: 0.5878
Validation set accuracy: 0.5342
Test set accuracy: 0.5327

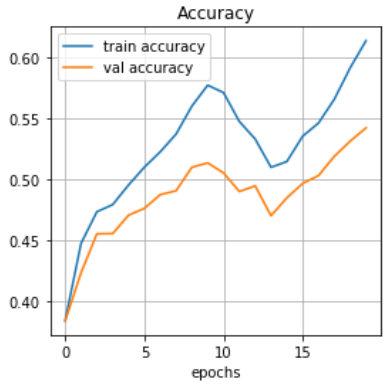




- **3-layer NN with Batch Normalization**

Train set accuracy: 0.61355
Validation set accuracy: 0.5422
Test set accuracy: 0.531

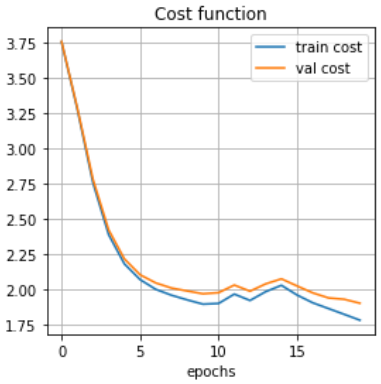


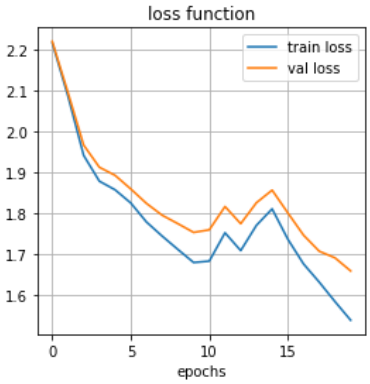


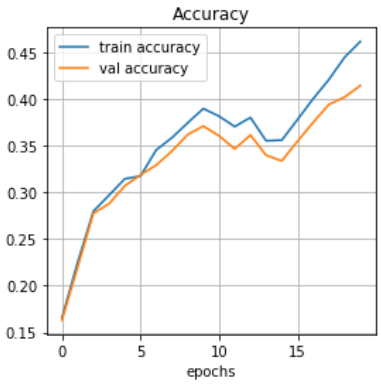
3) 9-layer network with He initialization

- **9-layer NN without Batch Normalization**

Train set accuracy: 0.4614
Validation set accuracy: 0.4142
Test set accuracy: 0.4227

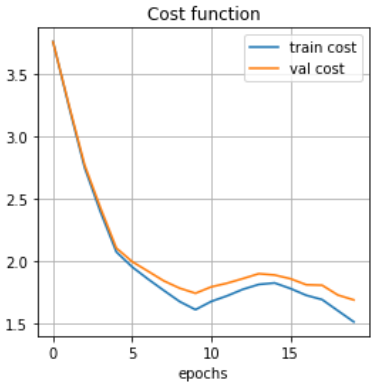


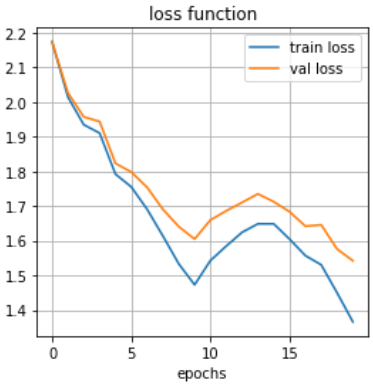


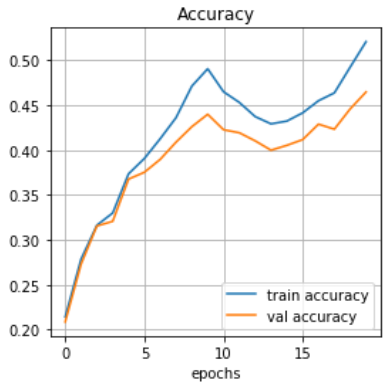


- **9-layer NN with Batch Normalization**

Train set accuracy: 0.52066
Validation set accuracy: 0.4646
Test set accuracy: 0.4624

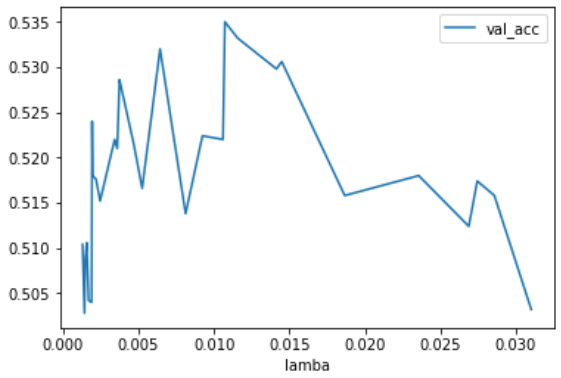






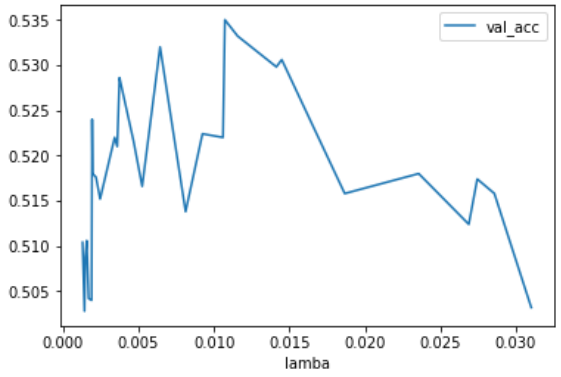
4) Regularization fine tuning(+advanced)

- **Coarsest ($\lambda \sim [10^{-5}\text{-}10^{-1}]$)**



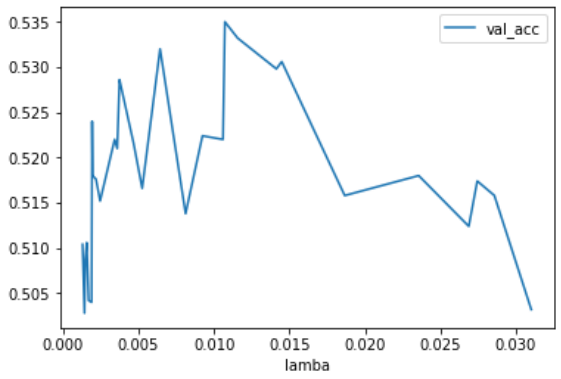
lamba	val_acc	train_acc
0.091176	0.4820	0.510756
0.066827	0.4866	0.518511
0.056512	0.4910	0.525756
0.000120	0.4966	0.582978
0.000846	0.4972	0.579156
0.000016	0.5010	0.585111
0.000070	0.5018	0.582778
0.000027	0.5022	0.584067
0.000210	0.5040	0.583911
0.000179	0.5044	0.583600
0.053900	0.5050	0.533956
0.000085	0.5052	0.582400
0.000285	0.5062	0.586911
0.000300	0.5070	0.584822
0.000013	0.5078	0.582444
0.022661	0.5084	0.558222
0.024070	0.5094	0.546578
0.000523	0.5098	0.586333
0.001493	0.5102	0.594111
0.001590	0.5130	0.590889
0.000142	0.5142	0.588733
0.000110	0.5152	0.582222
0.025500	0.5156	0.553933
0.017498	0.5180	0.567533
0.001831	0.5184	0.584956
0.015414	0.5206	0.571000
0.009585	0.5222	0.577111
0.006576	0.5248	0.589178
0.008412	0.5306	0.585867
0.016741	0.5328	0.573222

- **Coarse ($\lambda \sim [10^{-3}\text{-}10^{-1.7}]$)**



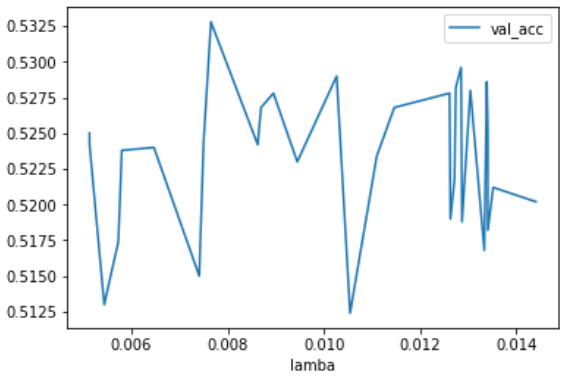
lamba	val_acc	train_acc
0.001154	0.5038	0.570711
0.001157	0.5072	0.591156
0.019080	0.5092	0.559289
0.001479	0.5094	0.586844
0.003280	0.5102	0.581689
0.002825	0.5120	0.574378
0.015074	0.5136	0.571933
0.001601	0.5142	0.593733
0.002102	0.5152	0.588289
0.004090	0.5152	0.583400
0.001632	0.5154	0.592956
0.017263	0.5156	0.561822
0.008324	0.5160	0.575911
0.015847	0.5172	0.563933
0.012052	0.5180	0.576178
0.002291	0.5180	0.592800
0.017035	0.5188	0.558844
0.001858	0.5190	0.591933
0.005226	0.5210	0.591400
0.006924	0.5214	0.591800
0.007946	0.5232	0.587911
0.017928	0.5240	0.569333
0.005880	0.5240	0.594556
0.005082	0.5250	0.588600
0.010812	0.5264	0.581422
0.006133	0.5272	0.592867
0.005447	0.5294	0.589489
0.007403	0.5300	0.586378
0.009606	0.5330	0.585000
0.005164	0.5332	0.597222

- **Fine (λ ~ [10^{-3}-$10^{-1.5}$])**



lamba	val_acc	train_acc
0.001419	0.5028	0.591622
0.031026	0.5032	0.540400
0.001889	0.5040	0.589800
0.001690	0.5042	0.583933
0.001472	0.5088	0.590244
0.001290	0.5104	0.586200
0.001564	0.5106	0.595000
0.026885	0.5124	0.552978
0.008117	0.5138	0.584156
0.002454	0.5152	0.599422
0.028581	0.5158	0.553533
0.018664	0.5158	0.564356
0.005241	0.5166	0.589511
0.027447	0.5174	0.551978
0.002182	0.5176	0.596911
0.001960	0.5180	0.597956
0.023567	0.5180	0.558378
0.003588	0.5210	0.592889
0.004669	0.5216	0.594489
0.010595	0.5220	0.580667
0.003417	0.5220	0.591578
0.009237	0.5224	0.580356
0.001922	0.5240	0.591422
0.003722	0.5278	0.592800
0.003741	0.5286	0.588689
0.014134	0.5298	0.577378
0.014491	0.5306	0.582333
0.006427	0.5320	0.585289
0.011568	0.5332	0.581311
0.010723	0.5350	0.586889

- **Finest (λ ~ [0.005-0.015])**



lamba	val_acc	train_acc
0.010546	0.5124	0.573711
0.005436	0.5130	0.581978
0.007411	0.5150	0.576844
0.013335	0.5168	0.564467
0.005729	0.5174	0.586444
0.013408	0.5182	0.573178
0.012872	0.5188	0.572156
0.012631	0.5190	0.570067
0.014409	0.5202	0.567978
0.013519	0.5212	0.572889
0.012716	0.5218	0.580311
0.009448	0.5230	0.581311
0.011100	0.5234	0.578156
0.005801	0.5238	0.582533
0.006469	0.5240	0.592089
0.005131	0.5242	0.590622
0.008626	0.5242	0.590578
0.007501	0.5244	0.588067
0.005130	0.5250	0.586089
0.013408	0.5252	0.570044
0.008695	0.5268	0.580378
0.011465	0.5268	0.580956
0.012612	0.5278	0.578889
0.008952	0.5278	0.589222
0.013045	0.5280	0.578378
0.012742	0.5282	0.573000
0.013388	0.5286	0.578778
0.010267	0.5290	0.581400
0.012852	0.5296	0.574467
0.007653	0.5328	0.585956

Best scores after 7 cycles ~ 90 epochs:

λ=0.005164

Train set accuracy: 0.61871

Validation set accuracy: 0.5372

Test set accuracy: 0.5355

λ=0.007653

Train set accuracy: 0.617022

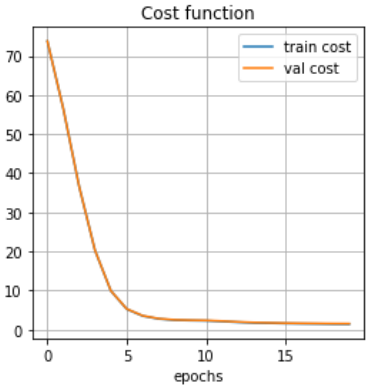
Validation set accuracy: 0.5426

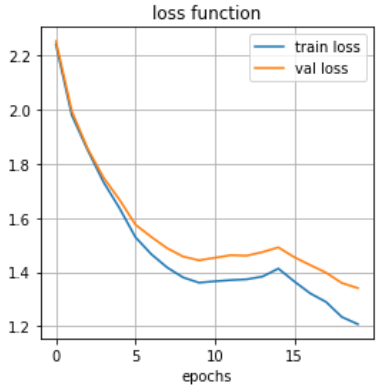
Test set accuracy: 0.5405

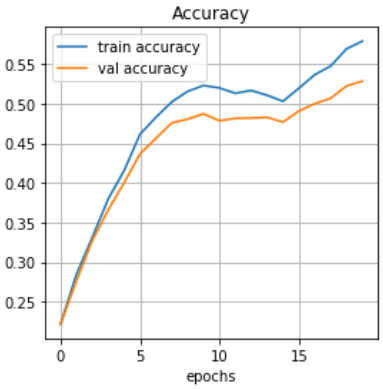
5.a) 3-layer network with Gauss initialization

- **3-layer NN without Batch Normalization (sig=1e-1)**

Train set accuracy: 0.578755
Validation set accuracy: 0.5282
Test set accuracy: 0.5285

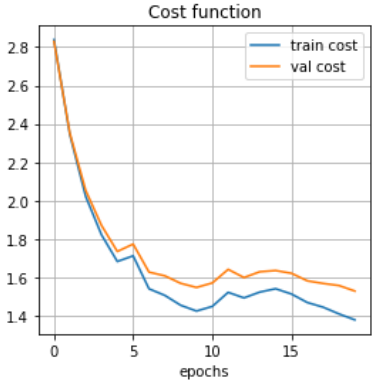


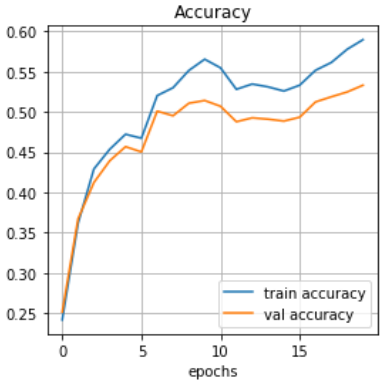




- **3-layer NN without Batch Normalization (sig=1e-3)**

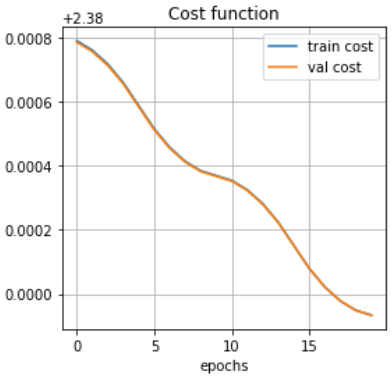
Train set accuracy: 0.5898
Validation set accuracy: 0.5332
Test set accuracy: 0.5276

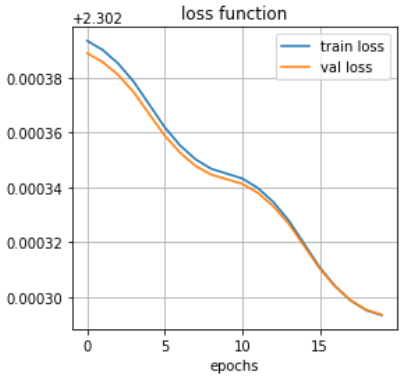


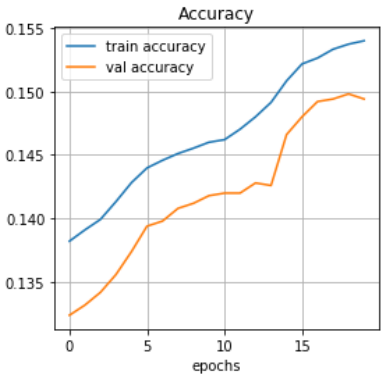


- **3-layer NN without Batch Normalization (sig=1e-4)**

Train set accuracy: 0.153977
Validation set accuracy: 0.1494
Test set accuracy: 0.1564

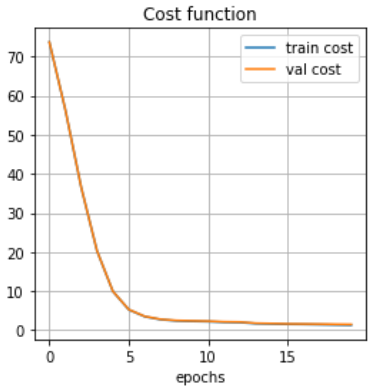


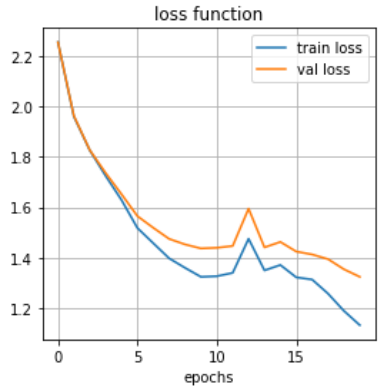


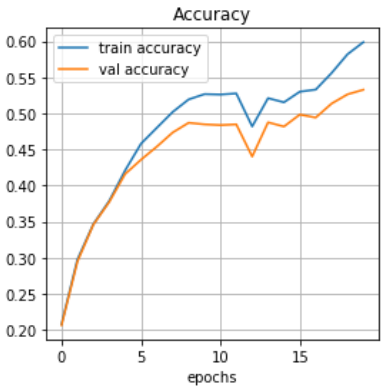


- **3-layer NN with Batch Normalization (sig=1e-1)**

Train set accuracy: 0.598711
Validation set accuracy: 0.5328
Test set accuracy: 0.523

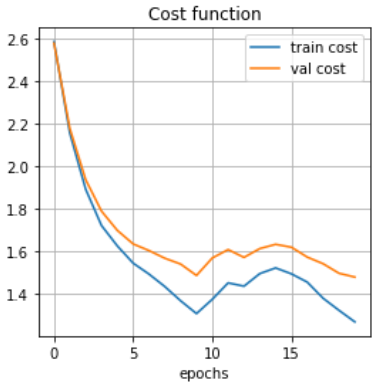


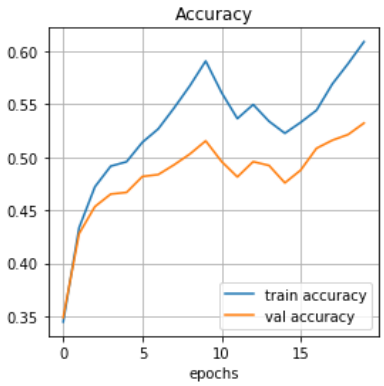




- **3-layer NN with Batch Normalization (sig=1e-3)**

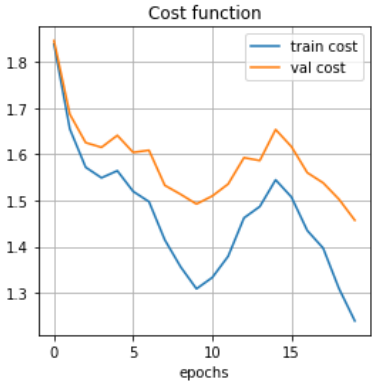
Train set accuracy: 0.608977
Validation set accuracy: 0.5322
Test set accuracy: 0.5297

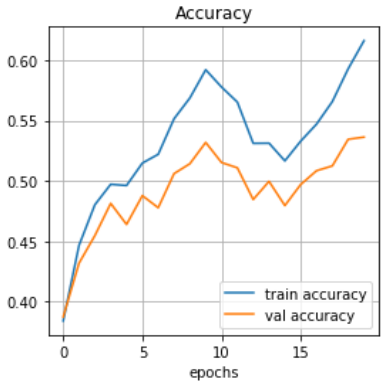




- **3-layer NN with Batch Normalization (sig=1e-4)**

Train set accuracy: 0.615933
Validation set accuracy: 0.5362
Test set accuracy: 0.5278

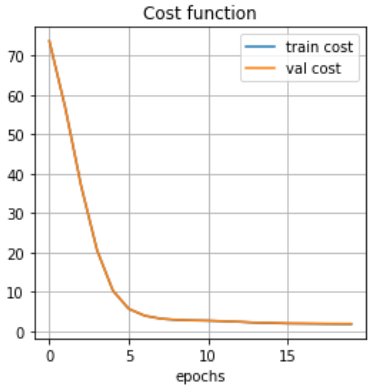


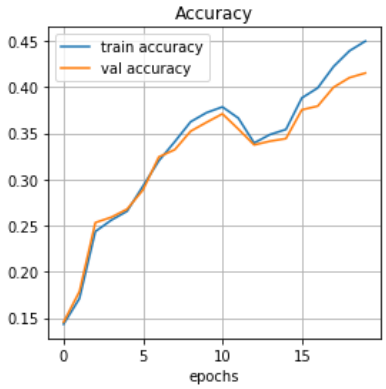


5.b) 9-layer network with Gauss initialization

- **9-layer NN without Batch Normalization (sig=1e-1)**

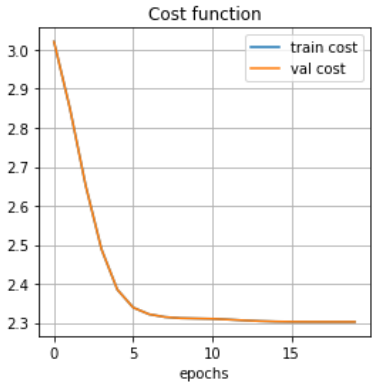
Train set accuracy: 0.449733
Validation set accuracy: 0.4152
Test set accuracy: 0.4215

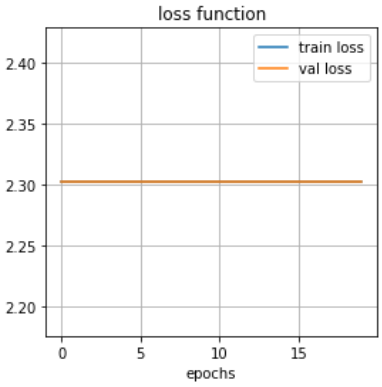


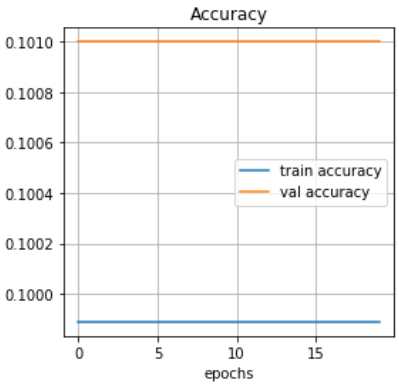


- **9-layer NN without Batch Normalization (sig=1e-3)**

Train set accuracy: 0.09988
Validation set accuracy: 0.101
Test set accuracy: 0.1

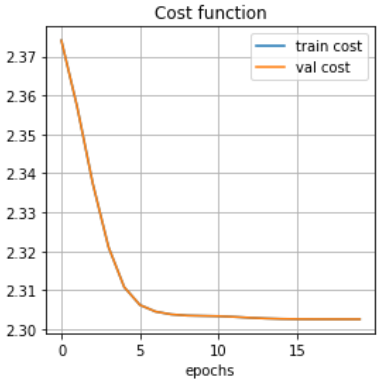


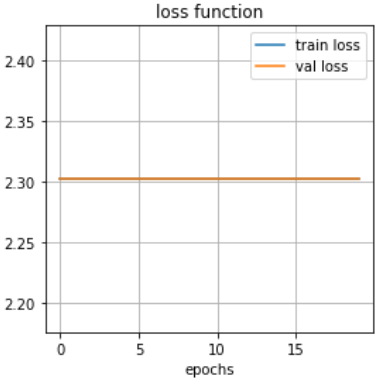


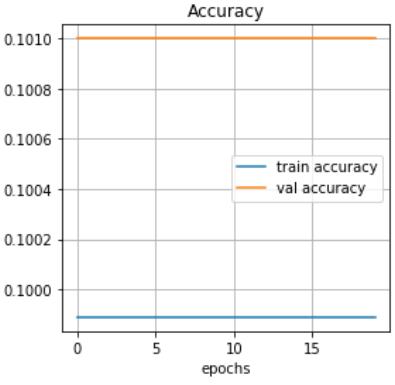


- **9-layer NN without Batch Normalization (sig=1e-4)**

Train set accuracy: 0.0998
Validation set accuracy: 0.101
Test set accuracy: 0.1

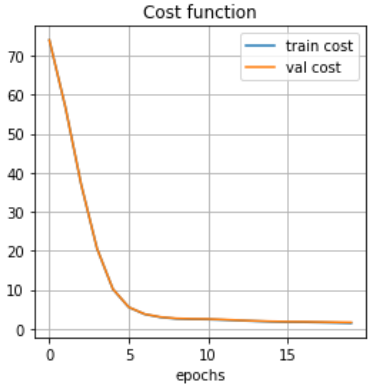


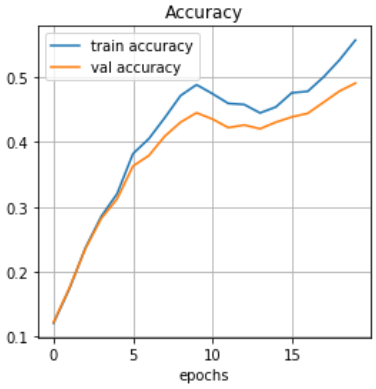




- **9-layer NN with Batch Normalization (sig=1e-1)**

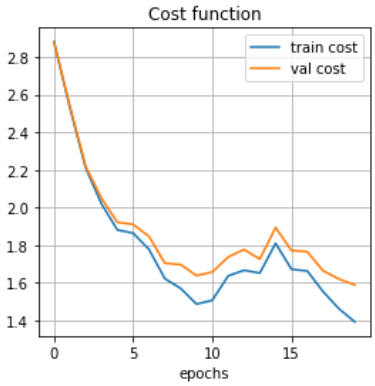
Train set accuracy: 0.5566
Validation set accuracy: 0.4904
Test set accuracy: 0.4948

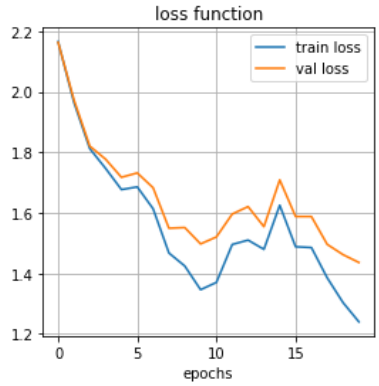


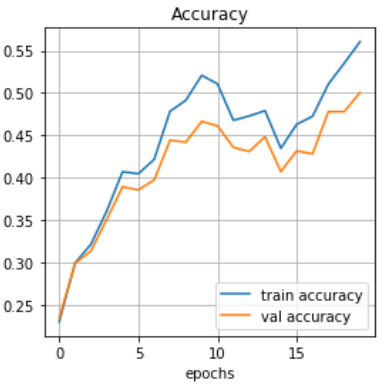


- **9-layer NN with Batch Normalization (sig=1e-3)**

Train set accuracy: 0.56048
Validation set accuracy: 0.5006
Test set accuracy: 0.4973

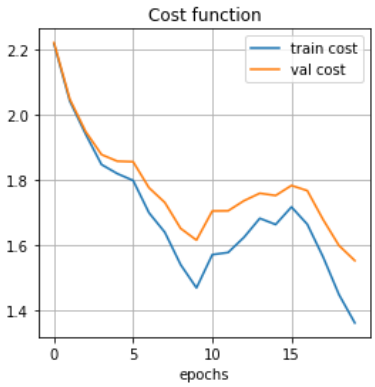


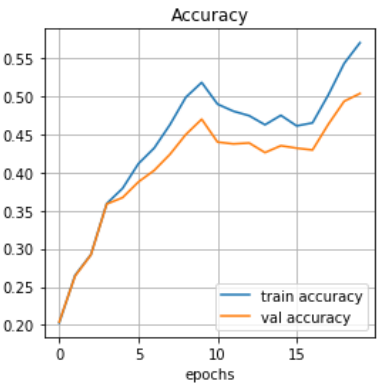




- **9-layer NN with Batch Normalization (sig=1e-4)**

Train set accuracy: 0.57073
Validation set accuracy: 0.504
Test set accuracy: 0.5076





In conclusion, we notice that Batch Normalization is an enhancement for neural networks which robustifies the training procedure against bad initializations. This is especially visible for deeper networks where the gradients die out much faster if the layers are initialized badly. This, supposedly, allows each layer to learn on a more stable distribution of inputs since variance and mean are less dependent on the transformations from the layers before.