The Forward-Forward Algorithm: Some Preliminary Investigations

G. Hinton

Presented by
Vishal V

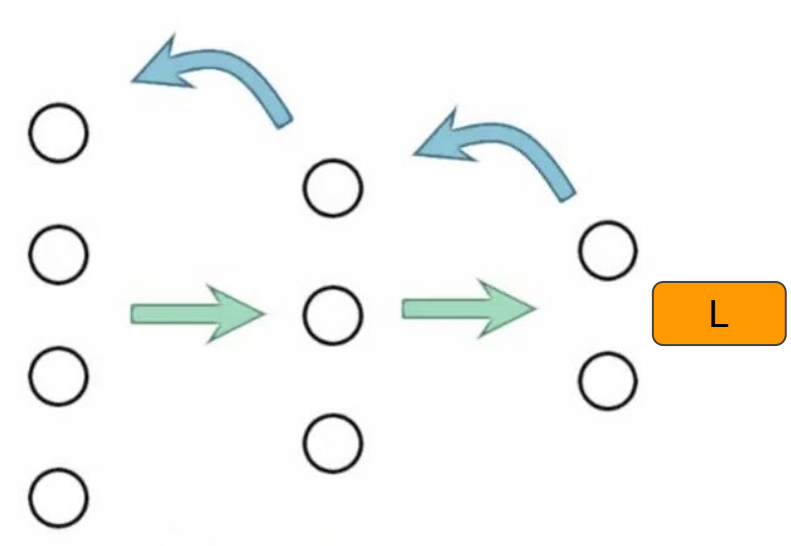
Table

What’s wrong with backprop

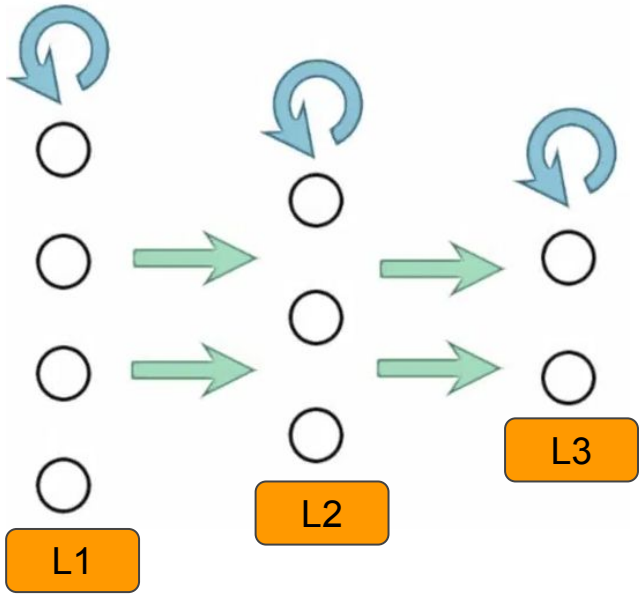
- Not Biologically Plausible
  - No evidence that brain store neural activities. For a backward pass like in backprop, storing activities would be necessary
  - Neurons do not wait for signals to pass through the entire brain to update connections. Brain needs to perform actions and learning in real-time without stopping

- Requires perfect knowledge of the forward pass
  - Weights and Biases can be updated only if we have total knowledge of forward and backward pass

Backpropagation



Forward-Forward



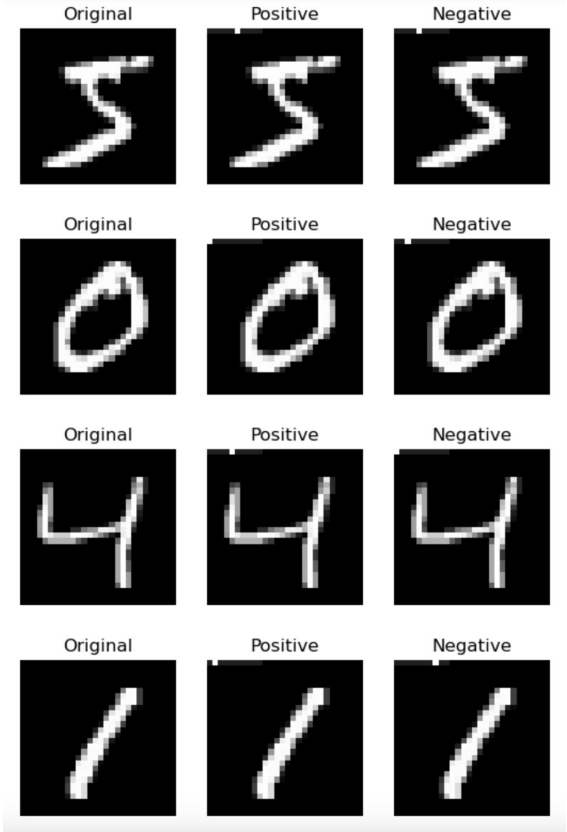
Loss function

$$\text{loss} = \frac{1}{N}\sum_{i=1}^{N}\left(\log\left(1+\exp\left(-g_{\text{pos},i}+\text{threshold}\right)\right)+\log\left(1+\exp\left(g_{\text{neg},i}-\text{threshold}\right)\right)\right)$$

- N is the total number of elements in the tensors g_{pos} and g_{neg}.
- $g_{\text{pos},i}$ and $g_{\text{neg},i}$ are the i-th elements of the tensors g_{pos} and g_{neg}, respectively.
- threshold is a scalar value.
- The log and exp functions apply element-wise.
- The $\sum$ notation indicates that we sum over all the elements resulting from the log operations, and then divide by N to compute the mean.

$$g = \frac{1}{M}\sum_{j=1}^{M} y_j^2$$

Positive and Negative data



For 0 original

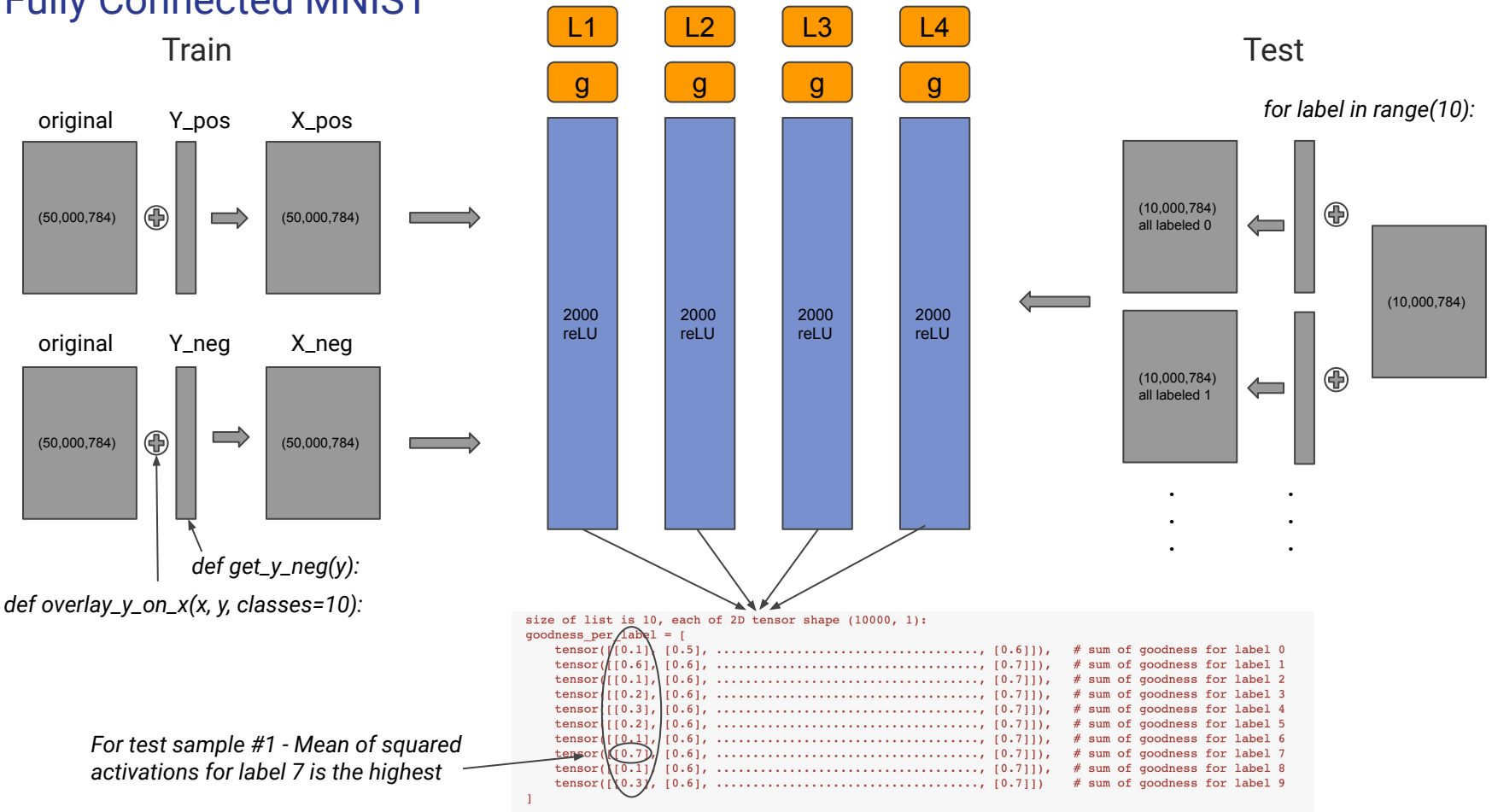
Positive data

- Correctly labeled data
- Should increase the mean of squared activities above a threshold

Negative data

- Incorrectly labeled data
- Should decrease the mean of squared activities above a threshold

Fully Connected MNIST



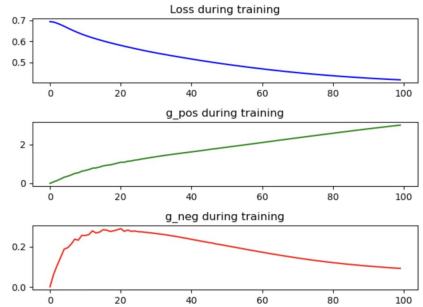

```
size of list is 10, each of 2D tensor shape (10000, 1):
goodness_per_label = [
    tensor([[0.1], [0.5], ..., [0.6]]),   # sum of goodness for label 0
    tensor([[0.6], [0.6], ..., [0.7]]),   # sum of goodness for label 1
    tensor([[0.1], [0.6], ..., [0.7]]),   # sum of goodness for label 2
    tensor([[0.2], [0.6], ..., [0.7]]),   # sum of goodness for label 3
    tensor([[0.3], [0.6], ..., [0.7]]),   # sum of goodness for label 4
    tensor([[0.2], [0.6], ..., [0.7]]),   # sum of goodness for label 5
    tensor([[0.1], [0.6], ..., [0.7]]),   # sum of goodness for label 6
    tensor([[0.7], [0.6], ..., [0.7]]),   # sum of goodness for label 7
    tensor([[0.1], [0.6], ..., [0.7]]),   # sum of goodness for label 8
    tensor([[0.3], [0.6], ..., [0.7]])    # sum of goodness for label 9
]
```

For test sample #1 - Mean of squared activations for label 7 is the highest

Layerwise Loss and goodness during train

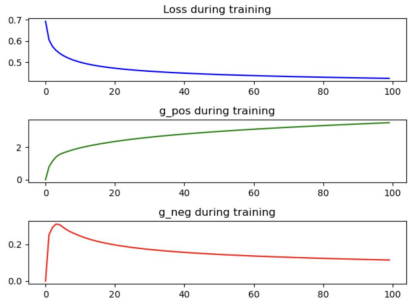
Threshold = 0

Layer 0



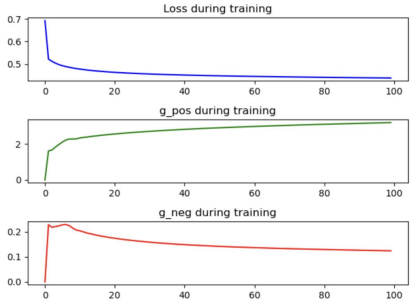

```
Loss at step 990: 0.4170325696468353
Loss at step 991: 0.41695263981819153
Loss at step 992: 0.41687288880348206
Loss at step 993: 0.41679325699806213
Loss at step 994: 0.4167138636112213
Loss at step 995: 0.41663458943367004
Loss at step 996: 0.41655561327934265
Loss at step 997: 0.4164767861366272
Loss at step 998: 0.4163980782032013
```

Layer 1



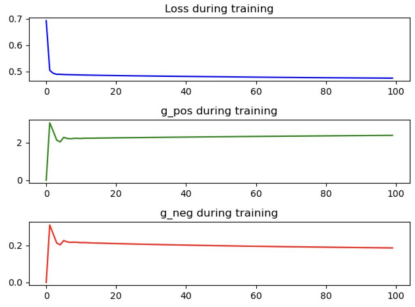

```
Loss at step 990: 0.42403945326805115
Loss at step 991: 0.4240147173404693 6
Loss at step 992: 0.4239899814128876
Loss at step 993: 0.4239652752876282
Loss at step 994: 0.4239405691623688
Loss at step 995: 0.4239159226417 5415
Loss at step 996: 0.4238913059234619
Loss at step 997: 0.4238667190074920 7
Loss at step 998: 0.4238421320915222
Loss at step 999: 0.4238175749778747 6
```

Layer 2




```
Loss at step 990: 0.4373338222503662
Loss at step 991: 0.4373199045658111 6
Loss at step 992: 0.4373059272766113 3
Loss at step 993: 0.437292009592056 3
Loss at step 994: 0.4372781217098236
Loss at step 995: 0.4372642382759094
Loss at step 996: 0.4372503757476806 6
Loss at step 997: 0.4372364878654488
Loss at step 998: 0.4372226595878601
Loss at step 999: 0.4372088313102722
```

Layer 3




```
Loss at step 990: 0.4747256934642792
Loss at step 991: 0.4747175574302673 3
Loss at step 992: 0.4747093617916107
Loss at step 993: 0.474701195955276 5
Loss at step 994: 0.4746930301189422 6
Loss at step 995: 0.4746849238872528
Loss at step 996: 0.4746767878532409 7
Loss at step 997: 0.4746686220169067 4
Loss at step 998: 0.474660485892894 9
Loss at step 999: 0.4746524095535278 3
```

Method	Layers	Epochs	Train Accuracy	Test Accuracy
FF	(2000, 2000, 2000, 2000)	1000 epochs	93.29%	93.18%

FF vs backprop results

Method	Layers	Epochs	Train Accuracy	Test Accuracy	Threshold
FF	(2000, 2000, 2000, 2000)	1000 epochs	93.29%	93.18%	0
FF	(2000, 2000, 2000, 2000)	1000 epochs	93.46%	93.58%	2
FF	(2000, 2000, 2000, 2000)	1000 epochs	93.45%	93.58%	3
Backprop	(2000, 2000, 2000, 2000)	15 epochs	93.73%	94.36%	-

FF in convnets CIFAR-10

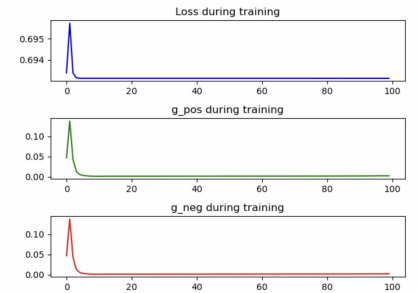
Network

```
# trial 7

self.layers = nn.Sequential(
Layer(3, 64, kernel_size=5, padding=2),  # Layer 0 with 5x5 receptive field
nn.ReLU(inplace=True),

Layer(64, 128, kernel_size=5, padding=2, stride=2),  # Layer 2 with 5x5 receptive field
nn.ReLU(inplace=True),

Layer(128, 256, kernel_size=5, padding=2, stride=2),  # Layer 4 with 5x5 receptive field
nn.ReLU(inplace=True),

Layer(256, 10, kernel_size=1, stride=1),  # Layer 6 with 1x1 receptive field
nn.ReLU(inplace=True),
)
```

Forward Pass in a layer

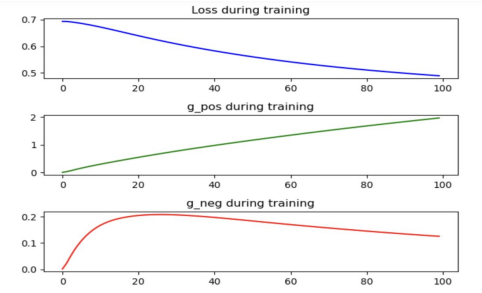
```
def forward(self, x):
    # L2 norm
    x_direction = x / (x.norm(2, 1, keepdim=True) + 1e-4)
    return self.relu(self.conv(x_direction))
```

Layer 0



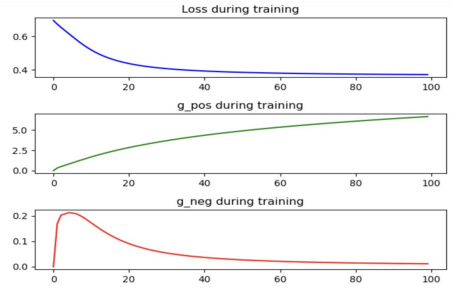

```
Loss at step 990: 0.6931461691856384
Loss at step 991: 0.6931461691856384
Loss at step 992: 0.6931461691856384
Loss at step 993: 0.6931461691856384
Loss at step 994: 0.6931461691856384
Loss at step 995: 0.6931461691856384
Loss at step 996: 0.6931461691856384
Loss at step 997: 0.6931461691856384
Loss at step 998: 0.6931461691856384
Loss at step 999: 0.6931461691856384
training layer:  1
```

Layer 2



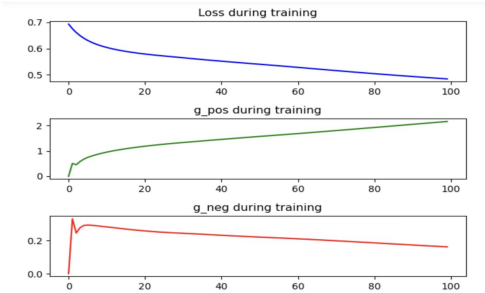

```
Loss at step 990: 0.4891962409019470
Loss at step 991: 0.4890972375869751
Loss at step 992: 0.4889984422882599
Loss at step 993: 0.4888997375965118
Loss at step 994: 0.4888012111186981
Loss at step 995: 0.4887027740478515
Loss at step 996: 0.4886045157909393
Loss at step 997: 0.4885063767433166
Loss at step 998: 0.4884083867073059
Loss at step 999: 0.4883104860782623
training layer:  3
training layer:  4
```

Layer 4




```
Loss at step 990: 0.3722074627876282
Loss at step 991: 0.372195959091865
Loss at step 992: 0.3721845499938965
Loss at step 993: 0.3721731007099151
Loss at step 994: 0.3721617162276306
Loss at step 995: 0.3721503913402557
Loss at step 996: 0.372139066457748
Loss at step 997: 0.3721277713757635
Loss at step 998: 0.3721165060997009
Loss at step 999: 0.3721052408218384
training layer:  5
training layer:  6
```

Layer 6

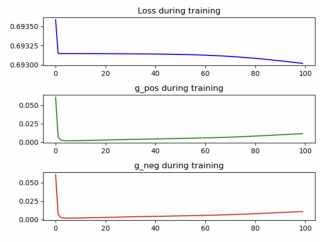



```
Loss at step 990: 0.4843975305557251
Loss at step 991: 0.4843003749847412
Loss at step 992: 0.4842032194137573
Loss at step 993: 0.4841061532497406
Loss at step 994: 0.4840091705322265
Loss at step 995: 0.4839122593402862
Loss at step 996: 0.4838154613971710
Loss at step 997: 0.4837186932563781
Loss at step 998: 0.4836220145225525
Loss at step 999: 0.4835254251956939
training layer:  7
```

Method	Layers	Epochs	Train Accuracy	Test Accuracy	Threshold
FF	trial #7	1000	9.90%	13.00%	0.002

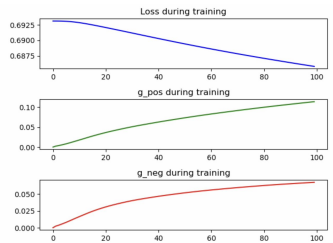
Deeper convnets

```
# trial 8

self.layers = nn.Sequential(
    Layer(3, 64, kernel_size=3, padding=1),  # Layer 0
    nn.ReLU(inplace=True),
    Layer(64, 64, kernel_size=3, padding=1),  # Layer 2
    nn.ReLU(inplace=True),

    nn.MaxPool2d(kernel_size=2, stride=2),  # Layer 4

    Layer(64, 128, kernel_size=3, padding=1),  # Layer 5
    nn.ReLU(inplace=True),
    Layer(128, 128, kernel_size=3, padding=1),  # Layer 7
    nn.ReLU(inplace=True),

    nn.MaxPool2d(kernel_size=2, stride=2),  # Layer 9

    Layer(128, 256, kernel_size=3, padding=1),  # Layer 10
    nn.ReLU(inplace=True),
    Layer(256, 256, kernel_size=3, padding=1),  # Layer 12
    nn.ReLU(inplace=True),
    Layer(256, 256, kernel_size=3, padding=1),  # Layer 14
    nn.ReLU(inplace=True),

    nn.MaxPool2d(kernel_size=2, stride=2),  # Layer 16

    Layer(256, 512, kernel_size=3, padding=1),  # Layer 17
    nn.ReLU(inplace=True),
    Layer(512, 512, kernel_size=3, padding=1),  # Layer 19
    nn.ReLU(inplace=True),
    Layer(512, 512, kernel_size=3, padding=1),  # Layer 21
    nn.ReLU(inplace=True),

    nn.MaxPool2d(kernel_size=2, stride=2),  # Layer 23

    Layer(512, 512, kernel_size=3, padding=1),  # Layer 24
    nn.ReLU(inplace=True),
    Layer(512, 512, kernel_size=3, padding=1),  # Layer 26
    nn.ReLU(inplace=True),
    Layer(512, 512, kernel_size=3, padding=1),  # Layer 28
    nn.ReLU(inplace=True)
)
```



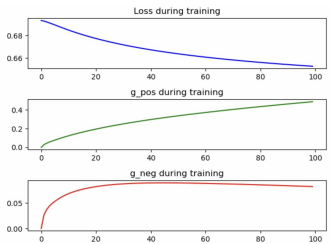

```
Loss at step 990: 0.6930199265480042
Loss at step 991: 0.6930195689201355
Loss at step 992: 0.6930192112922668
Loss at step 993: 0.6930188536643982
Loss at step 994: 0.6930183976827245
Loss at step 995: 0.6930179595947266
Loss at step 996: 0.6930176019668579
Loss at step 997: 0.6930172443389893
Loss at step 998: 0.6930168867111206
Loss at step 999: 0.6930164649486072
training layer: 1
training layer: 2
```

Layer 0



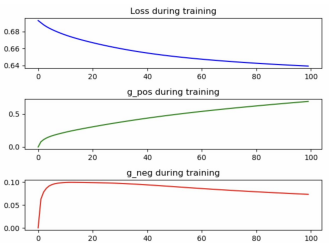

```
Loss at step 990: 0.6857848167419434
Loss at step 991: 0.6857782602310181
Loss at step 992: 0.6857717037200928
Loss at step 993: 0.6857650279998779
Loss at step 994: 0.6857584714889526
Loss at step 995: 0.6857519149780273
Loss at step 996: 0.6857454180717468
Loss at step 997: 0.6857388019561768
Loss at step 998: 0.6857323050498962
Loss at step 999: 0.6857256889343262
training layer: 3
training layer: 4
```

Layer 2



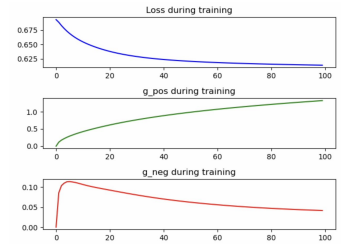

```
Loss at step 990: 0.6527884006500244
Loss at step 991: 0.6527727246284485
Loss at step 992: 0.6527569293975983
Loss at step 993: 0.6527412533760071
Loss at step 994: 0.6527255773544312
Loss at step 995: 0.6527097821235657
Loss at step 996: 0.6526942253112793
Loss at step 997: 0.6526784896850586
Loss at step 998: 0.6526628732681274
Loss at step 999: 0.6526473164558411
training layer: 6
training layer: 7
```

Layer 5



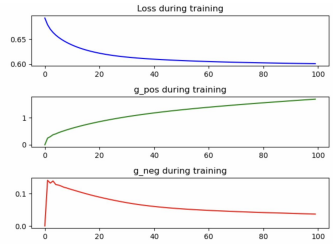

```
Loss at step 990: 0.6392584614726257
Loss at step 991: 0.6392377701892528
Loss at step 992: 0.6392225623130798
Loss at step 993: 0.6392075419425964
Loss at step 994: 0.6391924023628235
Loss at step 995: 0.6391773819923401
Loss at step 996: 0.6391623616218567
Loss at step 997: 0.6391475200653076
Loss at step 998: 0.6391324996948242
Loss at step 999: 0.6391175389289856
training layer: 8
training layer: 9
```

Layer 7



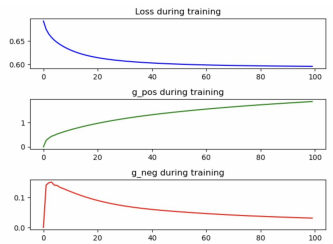

```
Loss at step 990: 0.6138436794281006
Loss at step 991: 0.6138360500335693
Loss at step 992: 0.6138284206390381
Loss at step 993: 0.6138208508491516
Loss at step 994: 0.6138131618499756
Loss at step 995: 0.613805592060852
Loss at step 996: 0.6137979626655579
Loss at step 997: 0.6137904524803162
Loss at step 998: 0.6137828230857849
Loss at step 999: 0.6137753129005432
training layer: 11
training layer: 12
```

Layer 10



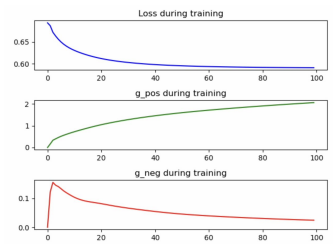

```
Loss at step 990: 0.6008676886558533
Loss at step 991: 0.6008612513542175
Loss at step 992: 0.600854754447937
Loss at step 993: 0.6008483767509460
Loss at step 994: 0.6008419394493103
Loss at step 995: 0.6008355021476746
Loss at step 996: 0.6008291244506836
Loss at step 997: 0.6008226871490479
Loss at step 998: 0.6008163690567017
Loss at step 999: 0.6008100509643555
training layer: 13
training layer: 14
```

Layer 12



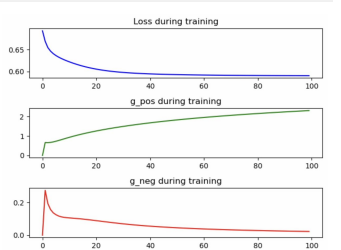

```
Loss at step 990: 0.5958105325698853
Loss at step 991: 0.5958060622215271
Loss at step 992: 0.595801591873169
Loss at step 993: 0.5957972121524048108
Loss at step 994: 0.5957927107810974
Loss at step 995: 0.5957882404327393
Loss at step 996: 0.5957838296890259
Loss at step 997: 0.5957792997360223
Loss at step 998: 0.5957749485969543
Loss at step 999: 0.5957705378532411
training layer: 15
training layer: 16
```

Layer 14



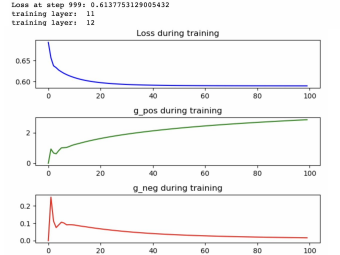

```
Loss at step 990: 0.5913243293762207
Loss at step 991: 0.5913212299346924
Loss at step 992: 0.5913181304931641
Loss at step 993: 0.5913150930404663085
Loss at step 994: 0.5913120508193397
Loss at step 995: 0.5913089513778687
Loss at step 996: 0.5913058519363403
Loss at step 997: 0.5913028717041016
Loss at step 998: 0.5912998318672218
Loss at step 999: 0.5912967920303345
training layer: 18
training layer: 19
```

Layer 17



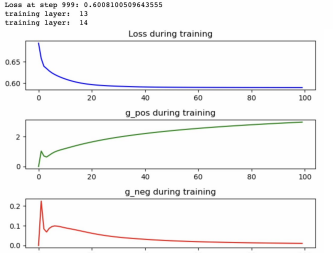

```
Loss at step 990: 0.5900912284851074
Loss at step 991: 0.5900892615318298
Loss at step 992: 0.5900874137878418
Loss at step 993: 0.5900855064439209
Loss at step 994: 0.5900836586952209
Loss at step 995: 0.5900817513465881
Loss at step 996: 0.5900799036026001
Loss at step 997: 0.5900781154632568
Loss at step 998: 0.5900762081146240
Loss at step 999: 0.5900743603706360
training layer: 20
training layer: 21
```

Layer 19



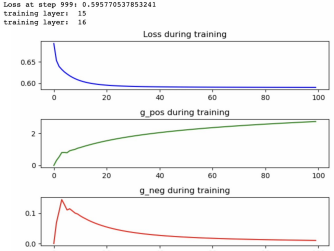

```
Loss at step 990: 0.5896961688995361
Loss at step 991: 0.5896955728530884
Loss at step 992: 0.5896949768066406
Loss at step 993: 0.5896943807601929
Loss at step 994: 0.5896939039230347
Loss at step 995: 0.5896932482719421
Loss at step 996: 0.5896927118301392
Loss at step 997: 0.5896921157836914
Loss at step 998: 0.5896915793418884
Loss at step 999: 0.5896909236907959
training layer: 22
training layer: 23
```

Layer 21



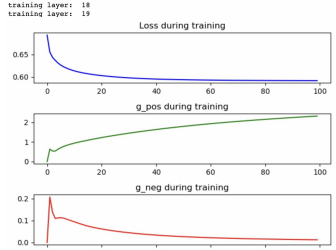

```
Loss at step 990: 0.5898072123527527
Loss at step 991: 0.5898066759109497
Loss at step 992: 0.5898061990737915
Loss at step 993: 0.5898056030273438
Loss at step 994: 0.5898050665855408
Loss at step 995: 0.5898045301437378
Loss at step 996: 0.5898040533065796
Loss at step 997: 0.5898035168647766
Loss at step 998: 0.5898029208183289
Loss at step 999: 0.5898023843765259
training layer: 25
training layer: 26
```

Layer 24




```
Loss at step 990: 0.5905284285545349
Loss at step 991: 0.5905275940895081
Loss at step 992: 0.5905267596244812
Loss at step 993: 0.5905260443687439
Loss at step 994: 0.5905252099037117
Loss at step 995: 0.5905244445800781
Loss at step 996: 0.5905236601829529
Loss at step 997: 0.5905228257179260
Loss at step 998: 0.5905220508575440
Loss at step 999: 0.5905212211799621
training layer: 27
training layer: 28
```

Layer 26

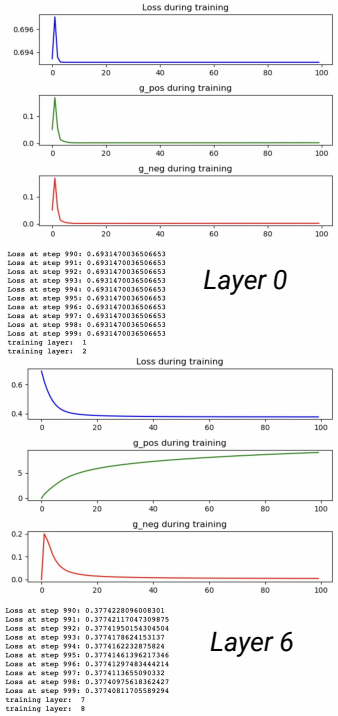



```
Loss at step 990: 0.5919664525985718
Loss at step 991: 0.5919650197029114
Loss at step 992: 0.5919636416435242
Loss at step 993: 0.5919619202613831
Loss at step 994: 0.5919604897499084
Loss at step 995: 0.5919589996337891
Loss at step 996: 0.5919575691223145
Loss at step 997: 0.5919560790061951
Loss at step 998: 0.5919547080993652
Loss at step 999: 0.5919531583786011
training layer: 29
```

Layer 28

Wider convnets

```
# trial 9

self.layers = nn.Sequential(
    Layer(3, 64, kernel_size=5, padding=2),  # Layer 0
    nn.ReLU(inplace=True),

    Layer(64, 128, kernel_size=5, padding=2, stride=2),  # Layer 2
    nn.ReLU(inplace=True),

    Layer(128, 256, kernel_size=5, padding=2, stride=2),  # Layer 4
    nn.ReLU(inplace=True),

    Layer(256, 512, kernel_size=5, padding=2, stride=2),  # Layer 6
    nn.ReLU(inplace=True),

    Layer(512, 1024, kernel_size=5, padding=2, stride=2),  # Layer 8
    nn.ReLU(inplace=True),

    Layer(1024, 2048, kernel_size=5, padding=2, stride=2),  # Layer 10
    nn.ReLU(inplace=True),

    Layer(2048, 10, kernel_size=1, stride=1),  # Layer 12
    nn.ReLU(inplace=True),
)
```



FF	trial #8	1000	9.90%	13.00%	0.002
FF	trial #9	1000	9.20%	10.00%	0.002

FF seems to be implausible for training convnets with the current loss and g function

Conclusion

- Explore other g function candidates - might solve for layerwise convnet training.
- Independent threshold params for each layer for better optimization.
- Try Biologically plausible activations - ReLU is used so far.

Thank You