Mini compte-rendu AS: implémentation et expériences du *Highway Network*

```
  ------------ HIGHWAY LAYER ------------
  local function Highway(inputSize)
    local input = nn.Identity()()
    local tGateLin = nn.Linear(inputSize, inputSize)(input):annotate{name = 'tGateLin'}
    local tGate = nn.Sigmoid()(tGateLin):annotate{name = 'transform'}
    local cGate = nn.AddConstant(1)(nn.MulConstant(-1)(tGate))
    local state = nn.Linear(inputSize, inputSize)(input)
    local output = nn.CAddTable()({
      nn.CMulTable()({state, tGate}),
      nn.CMulTable()({input, cGate})})
    return nn.gModule({input}, {output})
  end
  -----------------------------------------
```

```
------------ NETWORK ARCHITECTURE ------------
local net = nn.Sequential()
net:add(nn.Linear(opt.inputSize, opt.inputSize))
for n = 1, opt.nLayers-2 do
  net:add(nn.Tanh())
  net:add(nn.BatchNormalization(opt.inputSize))
  net:add(Highway(opt.inputSize))
end
net:add(nn.Tanh())
net:add(nn.BatchNormalization(opt.inputSize))
net:add(nn.Linear(opt.inputSize, opt.outputSize))
net:add(nn.SoftMax())
-----------------------------------------
```

```
------------ WEIGHT INITIALIZATION ------------
function weightsInit(m)
  local name = torch.type(m)
  if name:find('Linear') then
    m.weight:normal(0.0, 1e-3)
    m.bias:fill(0)
  elseif name:find('gModule') then
    for _, node in ipairs(m.forwardnodes) do
      if node.data.annotations.name == 'tGateLin' then
        node.data.module.bias:fill(-1)
      end
    end
  elseif name:find('BatchNormalization') then
    if m.weight then m.weight:normal(1.0, 0.02) end
    if m.bias then m.bias:fill(0) end
  end
end
-----------------------------------------
```

Les tests ont étés effectués avec des images de la base MNIST (6400 données d'apprentissage pour 1280) données de validation). Les données ont étés transformés de la manière suivante, (différentes combinaisons ont étés testés):

- **Chaque image** on) été centrées et réduite par rapport à la moyenne et l'écart type non biaisé de toutes de toutes les images (respectivement),
- Chaque variable (colonnes de la matrice des exemples) a été centrée et réduite également (on fait l'hypothèse que des pixels qui ne s'activent pas beaucoup aussi importants dans la décision finale pour la classification),
- Chaque exemple (ligne de la matrice) a été centrée et réduite (on fait l'hypothèse que ls changements de luminosité dans une image n'apportent pas d'information discriminante).

Voici ci-dessous les résultat d'un modèle initialisé avec les paramètres suivants:

```
------------ OPTIONS ------------
local opt = {
  printEvery = 50,
  nbTrain = 6400,
  nbValid = 200,
  maxIter = 3000,
  optimAlgo = 'adam',
  optimState = {learningRate = 1e-2}, weightDecay=1e-2},
  inputSize = 784,
  outputSize = 10,
  nLayers = 10,
  batchSize = 64,
  manualSeed = 123,
}
torch.manualSeed(opt.manualSeed)
---------------------------------------------
```

La première image représentation la matrice de confusion des données d'entraînement (confusionMatrix), gradWeightValues représente la norme 1 des poids pour chaque couche (1x10), et transformGateOutputs représente les valeurs de sortie des *transform gate* pour chaque variable, et pour chaque couche (10x784).