

1. Crie um Jupyter notebook para classificar o conjunto de dados do MNIST seguindo a seguinte arquitetura de rede.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_2 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 5, 5, 64)	0
conv2d_3 (Conv2D)	(None, 3, 3, 64)	36928
flatten_1 (Flatten)	(None, 576)	0
dense_1 (Dense)	(None, 64)	36928
dense_2 (Dense)	(None, 10)	650
Total params: 93,322		
Trainable params: 93,322		
Non-trainable params: 0		

No entanto, você deve experimentar diferentes funções de *loss* e diferentes otimizadores na sua rede. Acesse a documentação do Keras e escolha 3 funções de *loss* diferentes e 3 otimizadores diferentes.

Funções de *Loss*: <https://keras.io/losses/>

Otimizador: <https://keras.io/optimizers/>

Se a sua máquina permitir, aumente o tamanho do conjunto de dados, e ainda o número de épocas em relação ao que foi exposto em sala 😊 Faça uma análise sobre acurácia e o *loss* reportados no processo de treinamento do modelo para o conjunto de validação e treino em cada experimentação realizada.

O código visto em sala está em : <http://goo.gl/mxkS7h>

2. Escolha a melhor configuração da questão anterior para a Função de *Loss* e o otimizador, de acordo com a acurácia e o *loss*. Aplique o processo de *data augmentation* com *batch_size*=20 e adicione a rede uma camada de Dropout. Varie as probabilidades de os neurônios ficarem inativos

aplicando $p = 0.3, 0.4$ e 0.5 . Reporte a acurácia e o loss do conjunto validação e do conjunto de treino em cada um dos casos.

3. Escolha 3 classes do problema publicado pela StateFarm: Distracted Driver do Kaggle. Utilize 2000 instâncias para treino e 200 para validação para gerar um modelo capaz de classificar imagens em uma dessas 3 classes escolhidas. Aplique a técnica de *Feature Extraction*, utilizando a VGG16 com os pesos do ImageNet e treinando um novo modelo com 10 épocas e `batch_size` de 20. Reporte a acurácia e o loss, além de interpretar os resultados obtidos. A escolha do otimizador e da função *loss* fica com você. Reporte ainda se houve overfitting e em caso de overfitting, o que você faria? Dica: Use o Kernel do Kaggle para experimentar mais rapidamente, caso você não tenha GPU. Os dados do problema Distracted Driver: <http://goo.gl/dTd66m>.

O código visto em sala está em : goo.gl/bhVfRR