CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
INTELIGÊNCIA COMPUTACIONAL
PROF. ISMAEL SANTANA SILVA (ismaelsantana@cefetmg.br)

Prática 10 **– Rede Neural Convolucional**

Objetivo: Criação e aplicação de uma Rede Neural Convolucional (Convolutional Neural Network - CNN) utilizando o Framework TensorFlow.

Tarefa:

1. Abra, entenda o notebook ***CNN_CIFAR10_Prática.ipynb***;
2. Execute o notebook apenas com a CPU (não precisa esperar terminar) e depois utilizando GPU. Para isso, no Google Colab, acesse o menu: *Ambiente de execução >> Alterar o tipo de ambiente de execução*.
3. Reexecute o treinamento com 20 ou 30 épocas. Observe se há *overfitting* e analise o comportamento da validação.
4. Utilize `ImageDataGenerator` para aplicar rotações, inversões, deslocamentos e aumentos artificiais nos dados. Exemplo de código:

```
train_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
)

# No teste, só normalização (sem aumentos!)
test_datagen = ImageDataGenerator()

batch_size = 64 #O batch_size define quantas imagens são processadas por época
train_generator = train_datagen.flow(X_train, y_train_cat, batch_size=batch_size)
test_generator = test_datagen.flow(X_test, y_test_cat, batch_size=batch_size)

model.fit(
    train_generator,
    epochs=20,
    validation_data=test_generator
)
```

5. Troque o otimizador ***`sgd`*** ` por ` ***adam***` e `***rmsprop***` e compare os resultados. Explicação:

Otimizador	**Descrição breve**
sgd	Stochastic Gradiente descendente com taxa de aprendizado fixa.
sgd + momentum	Leva em conta a direção anterior para ganhar velocidade em vales rasos.
rmsprop	Adapta a *taxa de aprendizado* com base na variância do gradiente.
adam	Combina *Momentum + RMSProp*. É mais estável, rápido e exige menos ajuste manual.

Observação: "Estocástico" (Stochastic): significa que em vez de usar todo o conjunto de dados de uma vez, o algoritmo usa apenas uma amostra (ou mini-lote) a cada atualização. Isso torna o processo: (1) mais rápido, (2) com menos chance de *overfitting* e (3) com potencial para escapar de mínimos locais.

6. Alterar a arquitetura da CNN. Aumente a profundidade da rede com mais camadas. Acrescente convolucionais, ***BatchNormalization*** ou ***Dropout*** em locais estratégicos. Explicação:
   - Uma camada ***BatchNormalization()*** normalizar as ativações (ficam com média 0 e desvio 1).

- Comparar com uma MLP simples. Para isso, crie um modelo com camadas ***Flatten >> Dense*** sem convoluções e compare os resultados de acurácia.

7. Mostre 5 ou mais imagens que o seu melhor modelo classificou erradamente e informe a predição vs. classe real. Tente explicar porque o modelo errou.
8. Substitua CIFAR-10 pelo dataset ***cats_vs_dogs*** e crie uma CNN seguindo o que você aprendeu nessa prática. Observação:
    - Para carregar e pré-processar os dados utilize:

```
#Carregar o dataset Cats vs Dogs
(ds_train, ds_test), ds_info = tfds.load(
    'cats_vs_dogs',                      # Nome do dataset a ser carregado do TensorFlow Datasets (TFDS)
    split=['train[:80%]', 'train[80%:]'], # Divide o conjunto de treino em 80% para treino e 20% para teste
    with_info=True,                      # Retorna também metadados do dataset (número de classes, tamanho, etc)
    as_supervised=True                   # Retorna os dados no formato (imagem, label), ao invés de dicionários
)

# Redimensionamento e normalização
def preprocess(image, label):
    image = tf.image.resize(image, (128, 128))   # Redimensiona a imagem para 128x128 pixels
    image = tf.cast(image, tf.float32) / 255.0 #Converte a imagem para float e normaliza os valores entre [0,1]
    return image, label

# Aplica o preprocessamento no treino, cria batches de 32 imagens e pré-carrega 1 batch para otimizar
ds_train = ds_train.map(preprocess).batch(32).prefetch(1)

# Aplica o preprocessamento no teste, cria batches de 32 imagens e pré-carrega 1 batch para otimizar
ds_test = ds_test.map(preprocess).batch(32).prefetch(1)
```

    - Substitua a ultima camada da rede por uma ***sigmoid***: ***Dense(1, activation='sigmoid')***

9. **Entrega: dois notebooks Jupyter (.ipynb):** contendo o código alterado, documentado e com as análise solicitadas.