# Transfer Learning

Trabalho Prático II - Aprendizado de Máquina

## 1 Objetivo

O objetivo deste trabalho prático é de praticar o uso de *Convolutional Neural Networks* (CNN) e de *Transfer Learning*. Neste trabalho você irá comparar três estratégias de treino de CNNs, duas delas utilizando *Transfer Learning*, em um problema de classificação de imagens. A ideia é estudar como os pesos aprendidos em um modelo treinado para uma tarefa podem ser aproveitados em outra tarefa. Esta prática é bastante comum, principalmente considerando que o tempo para treinar modelos de *Deep Learning* pode ser muito alto, levando dias ou até semanas.

## 2 Tarefa

Neste trabalho você irá trabalhar com um conjunto de dados de imagens chamado CIFAR-10[1]. Este dataset contém 60000 imagens de tamanho 32x32, sendo 50000 de treino e 10000 de teste. Cada imagem pertence a uma das 10 classes, numeradas de 0 a 9. As classes são: airplane, automobile, bird, cat, deer, dog, frog, horse, ship e truck. O dataset pode ser baixado facilmente pelo Keras utilizando os seguintes comandos em Python:

```
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

A primeira etapa do trabalho consiste em dividir o conjunto de dados em duas partes A e B. Na parte B você irá selecionar todas as imagens pertencentes a 2 das 10 classes e na parte A ficarão as imagens das demais 8 classes. As classes selecionadas na parte B serão as de número $x$ e $y$, onde $x$ e $y$ são os dois últimos dígitos do seu número de matrícula. Se os dois últimos dígitos forem iguais, volte um digito a partir de $x$ até que os dois dígitos sejam distintos. Por exemplo, se seu número de matrícula é 2020387499, a parte B terá as classes de número 4 e 9.

Feito isto, você deverá treinar uma CNN para classificação das imagens usando os dados da parte A e a API do Keras para o TensorFlow. A CNN deverá ter quatro camadas de convolução e duas camadas de *Max Pooling* (não necessariamente nessa ordem) e duas camadas *Fully Connected* ao final. A função de ativação *ReLU* deve ser utilizada em todas as camadas que tem ativação exceto a última, que deve ter ativação *Softmax*. Também serão necessárias camadas de *Dropout* para a regularização do modelo. Você deverá ajustar os hiperparâmetros da rede de forma que o valor da função de perda (*loss function*) seja minimizado durante o treinamento e para controlar o overfitting. Para medir a generalização do modelo, utilize apenas o conjunto de teste (não é necessário usar validação cruzada) e a acurácia. O modelo deve ser treinado por 20 épocas (*epochs*).

Depois de treinar o modelo usando os dados da parte A, você deverá testar as três estratégias listadas abaixo para treinar um modelo usando os dados da parte B. Você deve utilizar a mesma

[1] https://www.cs.toronto.edu/~kriz/cifar.html

arquitetura da CNN treinada anteriormente e novamente treinar por 20 épocas. A acurácia deve ser utilizada para avaliar todos os modelos.

1. **Sem *Transfer Learning***: Treinar a rede do zero, inicializando os pesos aleatoriamente.
2. ***Fine-tuning* em uma camada**: Utilizar os pesos da rede treinada na parte A e realizar fine-tuning na última camada *Fully Connected*.
3. ***Fine-tuning* em duas camadas**: Utilizar os pesos da rede treinada na parte A e realizar fine-tuning nas duas camadas *Fully Connected*.

Ao final você deverá comparar os resultados das três estratégias, concluir qual obteve a melhor acurácia de teste e explicar qual foi o efeito de utilizar *Transfer Learning*. Para cada uma das 4 redes treinadas (uma da parte A e três da parte B), mostre, em um gráfico para cada, a convergência da função de perda e da acurácia de treino e de teste.

É recomendável utilizar uma GPU para treinar os modelos de CNN para diminuir o tempo gasto. Se você não possui acesso a uma GPU com Cuda, você pode utilizar o Google Colab (`https://colab.research.google.com/`). Nele é possível rodar um Notebook em nuvem utilizando GPU. Basta criar um notebook (que fica salvo no Google Drive) e no menu *Runtime*, na opção *Change Runtime type*, selecionar GPU como *Hardware Accelerator*.

# 3 Entrega

O trabalho deverá ser entregue no formato de Jupyter (IPython) notebook. O notebook deverá conter todo o código (devidamente comentado) necessário para executar os experimentos, a apresentação dos resultados por meios de texto, gráficos e tabelas, a explicação do que está sendo feito e a interpretação dos resultados. Apenas o notebook deve ser entregue. A organização e a clareza do notebook fazem parte da avaliação do trabalho. O monitor deverá ser capaz de reproduzir os experimentos apenas executando as células do notebook em ordem.

O notebook deverá ser submetido no Moodle até o dia 02/07 às 23:59 (apenas o arquivo .ipynb deve ser submetido). **Trabalhos submetidos fora do prazo não serão avaliados**. **Não haverá extensões do prazo**. **Se for detectado plágio, todos os alunos envolvidos receberão nota zero e serão encaminhados para a coordenação do curso**.

# 4 Critérios de Avaliação

- Implementação correta das redes, dos experimentos e da avaliação dos modelos (50%)
- Apresentação dos experimentos e dos resultados de forma clara, concisa e não ambígua (20%)
- Convergência dos modelos (o erro de treino é minimizado e o modelo está regularizado) (20%)
- Organização das informações apresentadas no notebook (10%)

Note que você não será avaliado pelos valores de acurácia obtidos, apenas o processo em si.