**Instituto Tecnológico de Aeronáutica – ITA**

**Inteligência Artificial para Robótica Móvel – CT-213**

**Professor: Marcos Ricardo Omena de Albuquerque Maximo**

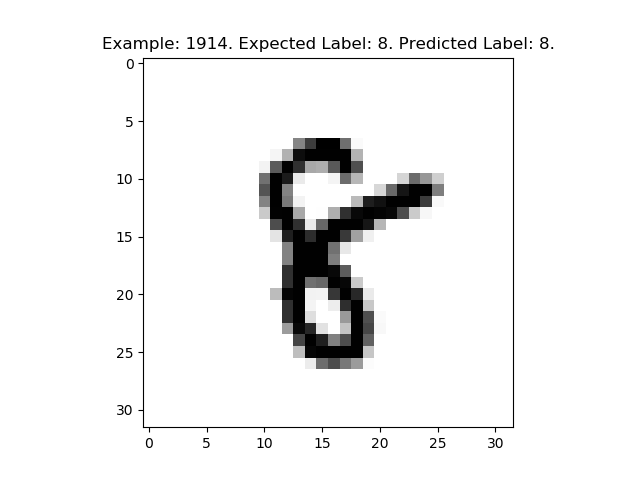
**Laboratório 8 (Aula 9) – Redes Neurais Convolucionais**

Para realizar esse laboratório, você precisará de um ambiente do Anaconda com as seguintes dependências:

* tensorflow
* tensorboard
* keras
* numpy
* matplotlib
* requests
* scikit-learn

1. **Introdução**

Nesse laboratório, seu objetivo é implementar, treinar e testar a rede neural LeNet-5 usando o *dataset* MNIST. O MNIST consiste num conjunto grande de imagens anotadas de dígitos decimais escritos à mão. Assim, você reproduzirá um trabalho clássico da Literatura de Redes Neurais Convolucionais (CNNs), que foi realizado originalmente por Yann LeCun. A Figura 1 apresenta uma das imagens do MNIST juntamente com o label esperado (de acordo com a anotação) e o label predito pela LeNet-5.



**Figura 1:** exemplo de dígito escrito à mão do *dataset* MNIST.

1. **Descrição do Problema**

O problema a ser resolvido nesse laboratório é implementar, treinar e testar a rede LeNet-5 no *dataset* MNIST. Para isso, você deve usar a *framework* Keras com backend Tensorflow. A Tabela 1 apresenta uma descrição da rede LeNet-5. A entrada da rede é (32, 32, 1), em que os números representam largura, altura e número de canais de cor da imagem, respectivamente. Para ajuda em como implementar cada uma dessas camadas usando Keras, recomenda-se verificar a Seção 6 (Dicas). Além disso, destaca-se que o aluno pode encontrar algumas interpretações mais modernas da LeNet-5 na Internet, mas pede-se para implementar a arquitetura dessa rede conforme descrito na Tabela 1.

| **# Camada** | **Tipo** | **Número de Filtros** | **Tamanho da Saída** | **Tamanho do Kernel** | **Stride** | **Activation Function** |
| --- | --- | --- | --- | --- | --- | --- |
| Entrada | Imagem | 1 | 32x32 | - | - | - |
| 1 | Conv2D | 6 | 28x28 | 5x5 | 1 | tanh |
| 2 | AveragePooling2D | 6 | 14x14 | 2x2 | 2 | - |
| 3 | Conv2D | 16 | 10x10 | 5x5 | 1 | tanh |
| 4 | AveragePooling2D | 16 | 5x5 | 2x2 | 2 | - |
| 5 | Conv2D | 120 | 1x1 | 5x5 | 1 | tanh |
| 6 | Dense (FC) | - | 84 | - | - | tanh |
| 7 | Dense (FC) | - | 10 | - | - | softmax |

**Tabela 1:** arquitetura da LeNet-5.

1. **Código Base**

O código base fornece diversos arquivos para facilitar o uso do NMIST e o treinamento da LeNet-5. Os seguintes arquivos foram fornecidos:

* download\_mnist.py: baixa da Internet o *dataset* MNIST.
* explore\_mnist.py: carrega o MNIST e exibe algumas imagens para ilustrar o *dataset*.
* run\_tensorboard.py: *script* auxiliar que facilita a execução do Tensorboard, uma ferramenta do Tensorflow para visualização de dados. Esse *script* é especialmente interessante caso esteja usando o ambiente proposto na disciplina (Anaconda com Pycharm). Você deve deixar esse *script* executando enquanto treina a rede para que possa verificar a evolução do treinamento no Tensorboard.
* lenet5.py: arquivo onde deve ser implementada a LeNet-5. Este é o único arquivo que você precisa editar nesse laboratório.
* train\_lenet5.py: treina a LeNet-5 usando o *training set* do MNIST. Salva o modelo treinado em arquivos.
* evaluate\_lenet5.py: avalia o modelo treinado no *test set* do MNIST.

1. **Tarefas**
   1. **Download do NMIST**

Execute o *script* download\_mnist.py. 4 arquivos devem ser baixados:

* [train-images-idx3-ubyte.gz](http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz): training set images (9912422 bytes).
* [train-labels-idx1-ubyte.gz](http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz): training set labels (28881 bytes).
* [t10k-images-idx3-ubyte.gz](http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz): test set images (1648877 bytes).
* [t10k-labels-idx1-ubyte.gz](http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz): test set labels (4542 bytes).
  1. **Exploração do NMIST**

Execute o *script* explore\_mnist.py para explorar o NMIST. Alguns dados sobre o *training e test sets* serão mostradas. Além disso, são mostradas algumas imagens aleatórias. Não há necessidade de incluir essas imagens no seu relatório.

* 1. **Implementação da LeNet-5**

Usando Keras, implemente a LeNet-5 no *script* lenet5.py de acordo com a arquitetura apresentada na Tabela 1. Há dicas na Seção 6 para te ajudar. Observações:

* Não há necessidade de se preocupar com regularização ou normalização para o correto treinamento da rede.
* A implementação da LeNet-5 é muito simples usando Keras. Serão poucas linhas de código.
* Como essa é uma rede muito clássica, há diversos lugares na Internet com implementações dela. Porém, recomendo que você tente implementar você mesmo a rede ao invés de procurar código já pronto. Além disso, perceba que a rede deve ser implementada exatamente conforme mostrado na Tabela 1.
  1. **Treinamento da LeNet-5**

Treine a LeNet-5 usando o *script* train\_lenet5.py. Leia o código do *script* e tente entender o que está sendo feito. Perceba que o *script* separa o *training set* original, deixando algumas imagens para um *validation set*.

Para verificar o treinamento da rede, você pode abrir o Tensorboard. Para isso, execute o script run\_tensorboard.py e então abra a url mostrado na linha de comando no seu browser.

Coloque imagens do Tensorboard mostrando a evolução do treinamento no seu relatório (pode tirar *print screen* dos gráficos do Tensorboard). Discuta os resultados obtidos.

* 1. **Avaliação da LeNet-5**

Avalie a LeNet-5 no *test set* usando o script evaluate\_letnet5.py. Além de exibir alguns exemplos aleatórios, incluindo o label anotado e o label predito pela LeNet-5, o *script* mostra alguns exemplos em que a rede errou a classificação. Inclua pelo menos um gráfico em que a predição da rede funcionou e outro em que ela errou a classificação no seu relatório. Comente os resultados obtidos.

1. **Entrega**

A entrega consiste do código e de um relatório, submetida através do Google Classroom. Modificações nos arquivos do código base são permitidas, desde que o nome e a interface dos scripts “main” não sejam alterados. A princípio, não há limitação de número de páginas para o relatório, mas pede-se que seja sucinto. O relatório deve conter:

* Breve descrição em alto nível da sua implementação.
* Figuras que comprovem o funcionamento do seu código.

Por limitações do Google Classroom (e por motivo de facilitar a automatização da correção), entregue seu laboratório com todos os arquivos num único arquivo **.zip** (**não** utilize outras tecnologias de compactação de arquivos) com o seguinte padrão de nome: “**<login\_email\_google\_education>\_labX.zip**”. Por exemplo, no meu caso, meu login Google Education é **marcos.maximo**, logo eu entregaria o lab 8 como “**marcos.maximo\_lab8.zip**”. **Não** crie subpastas para os arquivos da sua entrega, **deixe todos os arquivos na “raiz” do .zip**. Os relatórios devem ser entregues em formato **.pdf**.

1. **Dicas**

* Para criar uma camada de convolução 2D no Keras, faça:

model.add(layers.Conv2D(filters=nf, kernel\_size=(fx, fy), strides=(sx, sy), activation=activations.fun))

em que nf, fx, fy, sx, sy e fun configuram a camada.

* Para a primeira camada, também é importante definir o formato da entrada como um tensor de (32, 32, 1) (imagem de 32x32 e 1 canal de cor):

model.add(layers.Conv2D(filters=nf, kernel\_size=(fx, fy), strides=(sx, sy), activation=activations.fun, input\_shape=(32, 32, 1)))

* Para criar uma camada de *average pooling* no Keras, faça:

model.add(layers.AveragePooling2D(pool\_size=(px, py), strides=(sx, sy)))

em que px, py, sx e sy configuram a camada.

* Para definir a camada de transição entre as camadas convolucionais e as densas (completamente conectadas), a qual é a camada 5 na LeNet-5, faça o seguinte (isso é equivalente ao Conv2D indicado na camada 5):

model.add(layers.Flatten())

model.add(layers.Dense(units=num\_neurons, activation=activations.fun))