Aula 4
Redes Neurais Convolucionais #2

Eduardo L. L. Cabral

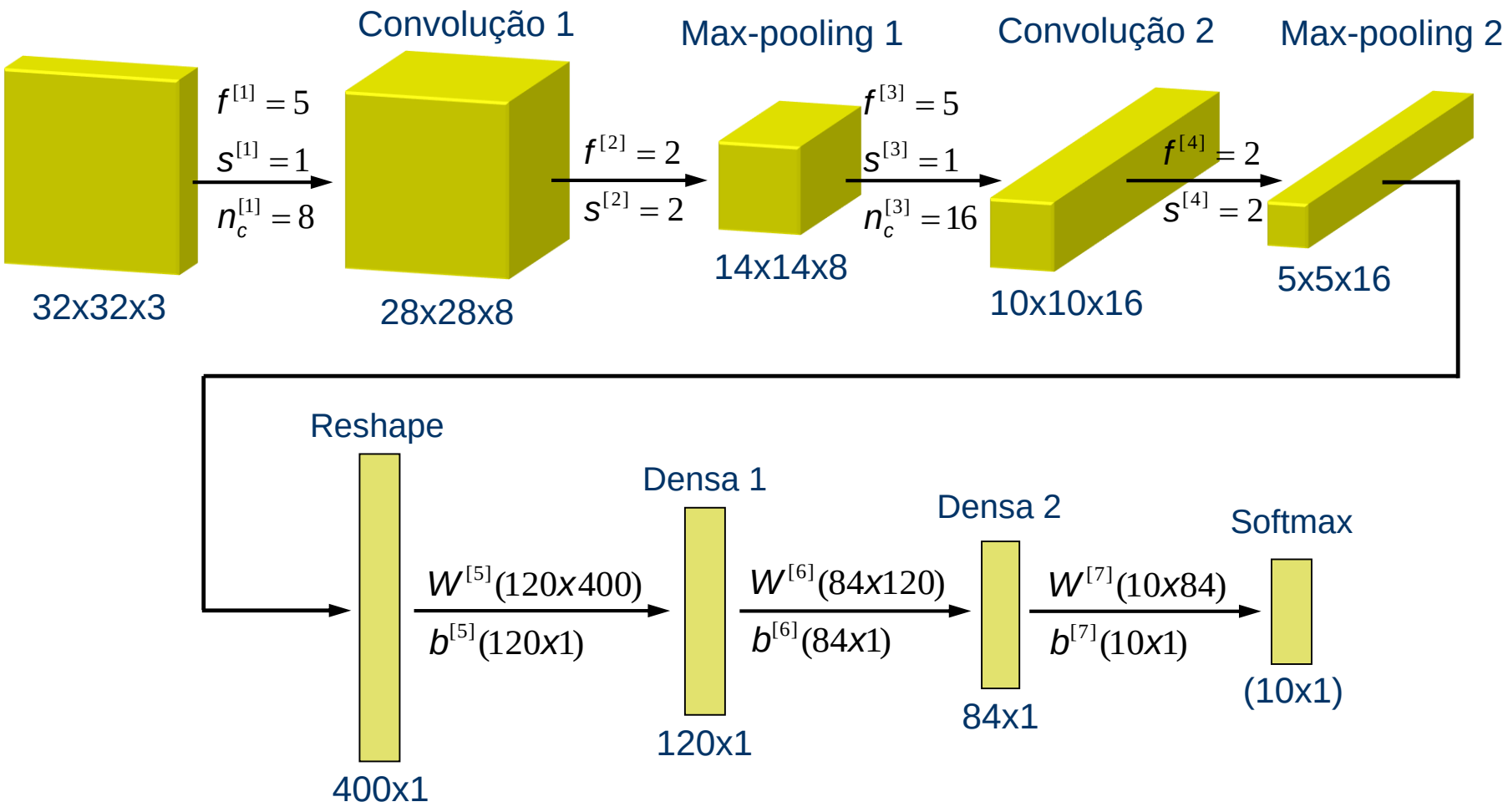
Objetivos

- Apresentar como configurar, compilar e treinar uma RNA convolucional com o Keras.
- Apresentar como visualizar as saídas das camadas convolucionais de uma RNA.
- Apresentar o que uma RNA convolucional realiza nas suas camadas.

Configuração de uma RNA no Keras

- A configuração de uma RNA convolucional usando o Keras segue os mesmos princípios usados para as redes com camadas densas.
- Como exemplo a rede LeNet-5 vista na aula 15 $\Rightarrow$ essa rede é usada para classificação multi-classe e possui a seguinte arquitetura:
  - Dimensão das imagens de entrada $\Rightarrow$ 32x32x3;
  - Camada convolucional $\Rightarrow$ $f^{[1]} = 5$, $p^{[1]} = 0$, $s^{[1]} = 1$, $n_c^{[1]} = 8$, ReLu;
  - Camada de “max-pooling” $\Rightarrow$ $f^{[2]} = 2$, $s^{[2]} = 2$;
  - Camada convolucional $\Rightarrow$ $f^{[3]} = 3$, $p^{[3]} = 0$, $s^{[3]} = 1$, $n_c^{[3]} = 16$, ReLu;
  - Camada de “max-pooling” $\Rightarrow$ $f^{[4]} = 2$, $s^{[4]} = 2$;
  - Camada de “flattening”;
  - Camada densa $\Rightarrow$ $n^{[5]} = 120$, ReLu;
  - Camada densa $\Rightarrow$ $n^{[6]} = 84$, Relu;
  - Camada softmax $\Rightarrow$ $n^{[7]} = 10$, softmax.

Arquitetura da RNA LeNet-5



Configuração de uma RNA no Keras

- O código a seguir apresenta como configurar as duas camadas convolucionais e as duas camadas de “pooling” da rede LeNet-5.

```
from keras import layers
from keras import models
rna = models.Sequential()
rna.add(layers.Conv2D(8, (5, 5), strides=1, padding='valid',
                      activation='relu', input_shape=(32, 32, 3)))
rna.add(layers.MaxPooling2D((2, 2)))
rna.add(layers.Conv2D(16, (5, 5), strides=1, padding='valid',
                  activation='relu'))
rna.add(layers.MaxPooling2D((2, 2)))
rna.summary()
```

  - Note que o número de filtros (número de canais) é o primeiro argumento passado na adição da camada convolucional.
  - Na camada de “pooling” o primeiro valor é a dimensão da janela e o segundo é o “stride”.

Configuração de uma RNA no keras

- O padrão das camadas convolucionais no Keras é $s = 1$ e $p = 0$ (“valid”), assim, se esse for o caso não precisaria incluir essas info.
- A execução desses comandos resulta na configuração da RNA com a arquitetura mostrada no quadro a seguir.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 8)         608
_________________________________________________________________
max_pooling2d_1 (MaxPooling2) (None, 14, 14, 8)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 16)        3216
_________________________________________________________________
max_pooling2d_2 (MaxPooling2) (None, 5, 5, 16)         0
=================================================================
Total params: 3,824
Trainable params: 3,824
Non-trainable params: 0
```

Configuração de uma RNA no Keras

- Falta adicionar na RNA a camada de “flattening” e as camadas densas. O código a seguir mostra como fazer essa etapa.

```
rna.add(layers.Flatten())
rna.add(layers.Dense(120, activation='relu'))
rna.add(layers.Dense(84, activation='relu'))
rna.add(layers.Dense(10, activation='softmax'))
rna.summary()
```

  - O redimensionamento da saída da última camada convolucional para transformar o tensor 3D em um vetor é realizada pela camada `Flatten`.

- A execução desse código resulta na RNA mostrada no quadro a seguir.

Configuração de uma RNA no Keras

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 28, 28, 8)         608
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 14, 14, 8)         0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 10, 10, 16)        3216
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 5, 5, 16)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_3 (Dense)              (None, 120)               48120
_________________________________________________________________
dense_4 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_5 (Dense)              (None, 10)                850
=================================================================
Total params: 62,958
Trainable params: 62,958
Non-trainable params: 0
```

Compilação e treinamento

- A compilação e treinamento de uma RNA convolucional no Keras é realizada exatamente como nas redes com somente camadas densas.
- O quadro a seguir mostra um exemplo de como compilar e treinar a rede configurada.

```
rna.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
rna.fit(train_images, train_labels, epochs=5, batch_size=64)
```

  - Nesse caso é utilizado o método de otimização RMSprop, com 5 épocas de treinamento e tamanho da mini-batelada de 64.
  - Não são usados exemplos de validação.

Visualização do aprendizado

- A visualização das saídas das camadas convolucionais de uma RNA é importante para entender o que de fato a rede aprendeu e o que está fazendo para processar os dados.
- Existem várias formas de visualização e interpretação dos resultados de uma RNA convolucional. As três formas mais utilizadas são:
  - Visualização das saídas das camadas intermediárias da rede $\Rightarrow$ útil pra entender como as sucessivas camadas convolucionais transformam suas entradas e entender o que realiza cada filtro da camada;
  - Visualização dos filtros $\Rightarrow$ útil para entender cada padrão que a rede é capaz de detectar e conhecer os filtros que a rede aprendeu;
  - Visualização de mapas de calor ou de ativação de classes $\Rightarrow$ útil para entender que parte de uma imagem é identificada como sendo de uma dada classe, de forma a permitir localizar objetos nas imagens.

Visualização das saídas das camadas

- A visualização das ativações das camadas convolucionais intermediárias de uma RNA consiste em mostrar os mapas de características que são produzidas pelos filtros para uma dada entrada da rede.
- Essa visualização permite verificar como uma dada entrada é decomposta pelos diversos filtros aprendidos pela RNA.
- A saída de cada camada convolucional é um tensor 3D, onde se tem um número de “imagens” igual ao número de canais (filtros) da camada.
- Cada canal (filtro) de um camada convolucional codifica de forma independente características diferentes de forma que a melhor forma de visualização é fazer o gráfico da saída de cada filtro como se fosse uma imagem 2D.

Visualização das saídas das camadas

- O primeiro passo é carregar uma RNA já treinada e armazenada.

```
# Importa funções
from keras.models import load_model

# Carrega rna salva e apresenta sua arquitetura
rna = load_model('rna.h5')
rna.summary()
```

- Os próximos passos são:
  - Gerar um modelo do Keras que permite múltiplas saídas;
  - rna.h5 é o nome do arquivo onde se encontra a RNA;
  - Obter uma imagem de teste $\Rightarrow$ obviamente que essa imagem deve ser processada da mesma forma como foram processadas as imagens usadas no treinamento, ou seja, deve-se normalizá-la.

Visualização das saídas das camadas

- Para visualizar as saídas das camadas de uma RNA deve-se criar um modelo Keras que recebe uma imagem como entrada e gera como saída as ativações das camadas que se deseja visualizar $\Rightarrow$ o Keras possui uma classe de modelos para fazer isso: Keras Class Model.
- Esse tipo de modelo é equivalente ao modelo sequencial de uma RNA, mas pode ter múltiplas entradas e múltiplas saídas.
- Observa-se que no caso geral uma RNA pode ter qualquer número de entradas e saídas.
- Esse tipo de modelo é criado usando dois argumentos $\Rightarrow$ uma lista de tensores de entrada e uma lista de tensores de saída.

Visualização das saídas das camadas

- O código abaixo mostra um exemplo de como criar um modelo desse tipo.

```
# Importa classe de modelos do keras
from keras import models
# Define as saídas como sendo as ativações das 4 primeiras
camadas da RNA
camadas_saidas = [layer.output for layer in rna.layers[:4]]
# Cria o modelo que retorna as ativações das camadas, dada uma
entrada
rna_ativacoes = models.Model(inputs=rna.input,
                             outputs=camadas_saidas)
```

- Quando esse modelo recebe uma imagem de entrada, ele retorna as ativações das camadas da RNA original.
- No caso dessa RNA tem-se uma entrada e quatro saídas (uma saída para cada conjunto de ativações de uma camada).

Visualização das saídas das camadas

- A imagem usada como entrada dessa nova RNA deve ser um tensor de mesmo tamanho que o usado na RNA original.
- Uma imagem colorida tem 3 eixos (altura, largura, cor) e o tensor de entrada da RNA tem 4 eixos (exemplo, altura, largura, cor) $\Rightarrow$ portanto, deve-se incluir um quarto eixo na imagem antes dela ser usada como entrada da RNA.
- O código abaixo mostra como incluir esse novo eixo em uma imagem colorida.

```
# Escolhe 5ª imagem do conjunto de teste e transforma em um
tensor com mesmo numero de eixos do tensor de entrada da RNA
index = 4
imagem = np.expand_dims(X_test[index], axis=0)
```

Visualização das saídas das camadas

- O próximo passo é executar a nova RNA em modo de predição.
- O código abaixo mostra como fazer isso e também como fazer o gráfico da saída do 5º filtro da primeira camada.

```
# Calcula saídas da RNA (saídas das 4 primeiras camadas)
ativacoes = rna_ativacoes.predict(imagem)
# Guarda saída da 1ª camada para fazer gráfico
ativacao_first_layer = ativacoes[0]
print(ativacao_first_layer.shape)
# Faz o gráfico da saída do 5º filtro da primeira camada
plt.matshow(ativação_first_layer[0, :, :, 4], cmap='viridis')

# Resultado do comando print
(1, 28, 28, 8)
```

- A saída da 1ª camada convolucional é um mapa de características de dimensão 28x28 com 8 canais.

RNAs convolucionais - conclusão

- Em geral as primeiras camadas de uma RNA convolucional agem como uma coleção de detectores de vários tipos de bordas.
- Nas primeiras camadas a ativações contém quase toda a informação presente na imagem original.
- Na medida que avançamos para dentro da rede, as ativações se tornam mais abstratas e com menor significado visual e começam a codificar características de alto nível.
- Características de níveis mais alto contém menos informação visual e mais informações relacionadas com a tarefa a ser realizada.
- A não ativação de filtros aumenta com a profundidade da camada $\Rightarrow$ na 1ª camada praticamente todos os filtros são ativados, mas nas camadas mais profundas menos filtros ficam ativos.

RNAs convolucionais - conclusão

- Quando um filtro não é ativado por uma imagem significa que o padrão codificado por aquele filtro não está presente naquela imagem.
- Uma característica importante das RNAs convolucionais deep learning é que as características aprendidas pelas suas camadas se tornam cada vez mais abstratas com a profundidade da camada.
- As ativações de camadas mais profundas contém menos informação visual e mais informação sobre a tarefa a ser realizada.
- Uma RNA deep learning age efetivamente como um destilador de informação, onde dados brutos são repetidamente transformados de forma que informações irrelevantes são descartadas e informações importantes são ressaltadas e refinadas.

RNAs convolucionais - conclusão

- A forma que uma RNA convolucional deep learning opera é análoga à forma como os seres humanos e animais percebem o mundo $\Rightarrow$ após observar uma cena por alguns segundos uma pessoa pode lembrar quais objetos estavam presentes (bicicleta, árvore, carro etc), mas não é capaz de lembrar de aspectos específicos desses objetos.
- O cérebro humano é treinado para abstrair as imagens vistas e transformá-las em conceitos visual de alto nível, descartando detalhes visuais irrelevantes.
- No próximo trabalho você irá verificar pessoalmente essas conclusões.