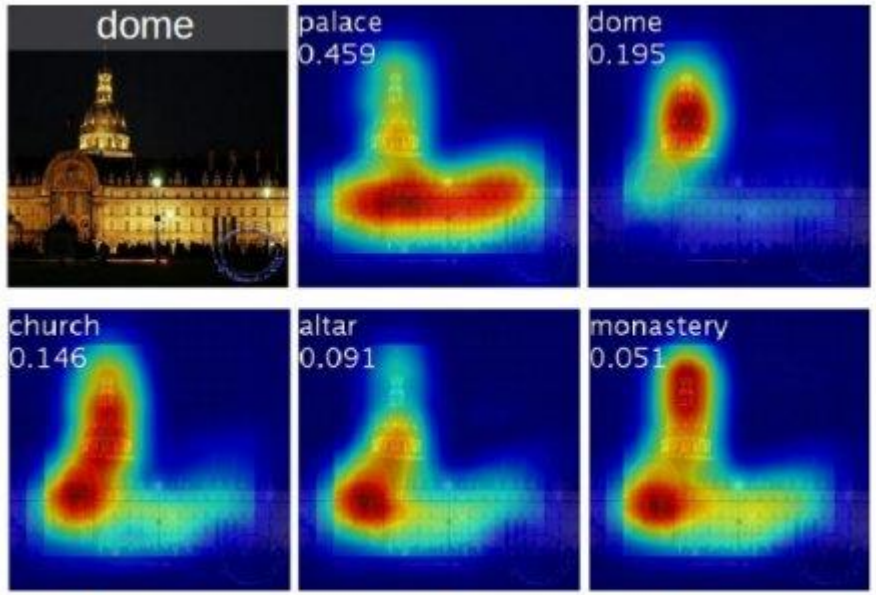

بسم الله الرحمن الرحيم

مصوّرسازی

Visualization

نمایش نقشه‌های حرارتی (heatmaps)

- نقشه فعالیت کلاس (Class Activation Map)
- نشان می‌دهد که هر مکان با توجه به کلاس مورد بررسی چقدر اهمیت دارد
- برای درک اینکه یک شبکه بر اساس کدام بخش از یک تصویر به تصمیم نهایی رسیده است

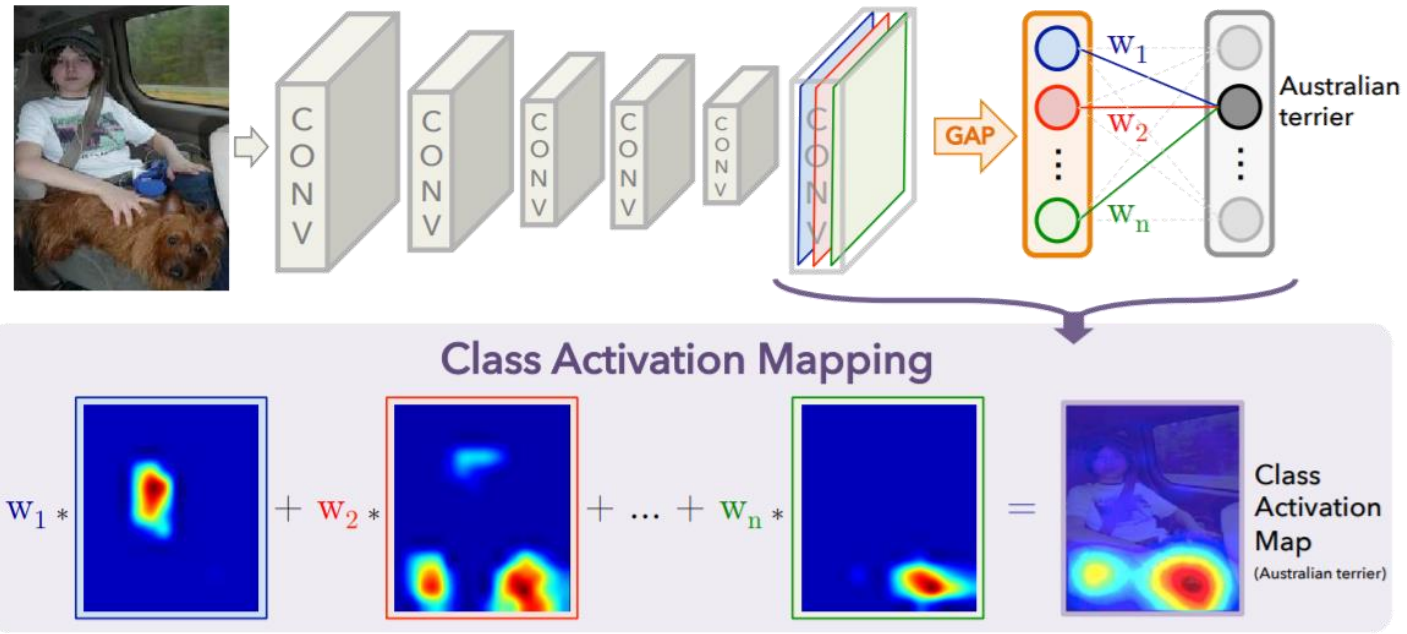


Class activation maps of top 5 predictions

Class activation maps for one object class

https://github.com/zhoubolei/CAM

CAM



- $f_k(x,y)$ خروجی آخرین لایه کانولوشنی قبل از GAP است
- S_c احتمال (غیرنرمالیزه) پیش‌بینی شده برای کلاس c است (قبل از Softmax)

$$M_c(x,y) = \sum_k w_k^c f_k(x,y)$$

$$F_k = \sum_{x,y} f_k(x,y)$$

$$S_c = \sum_k w_k^c F_k = \sum_k w_k^c \sum_{x,y} f_k(x,y) \sum_k \sum_{x,y} w_k^c f_k(x,y) \sum_{x,y} M_c(x,y)$$

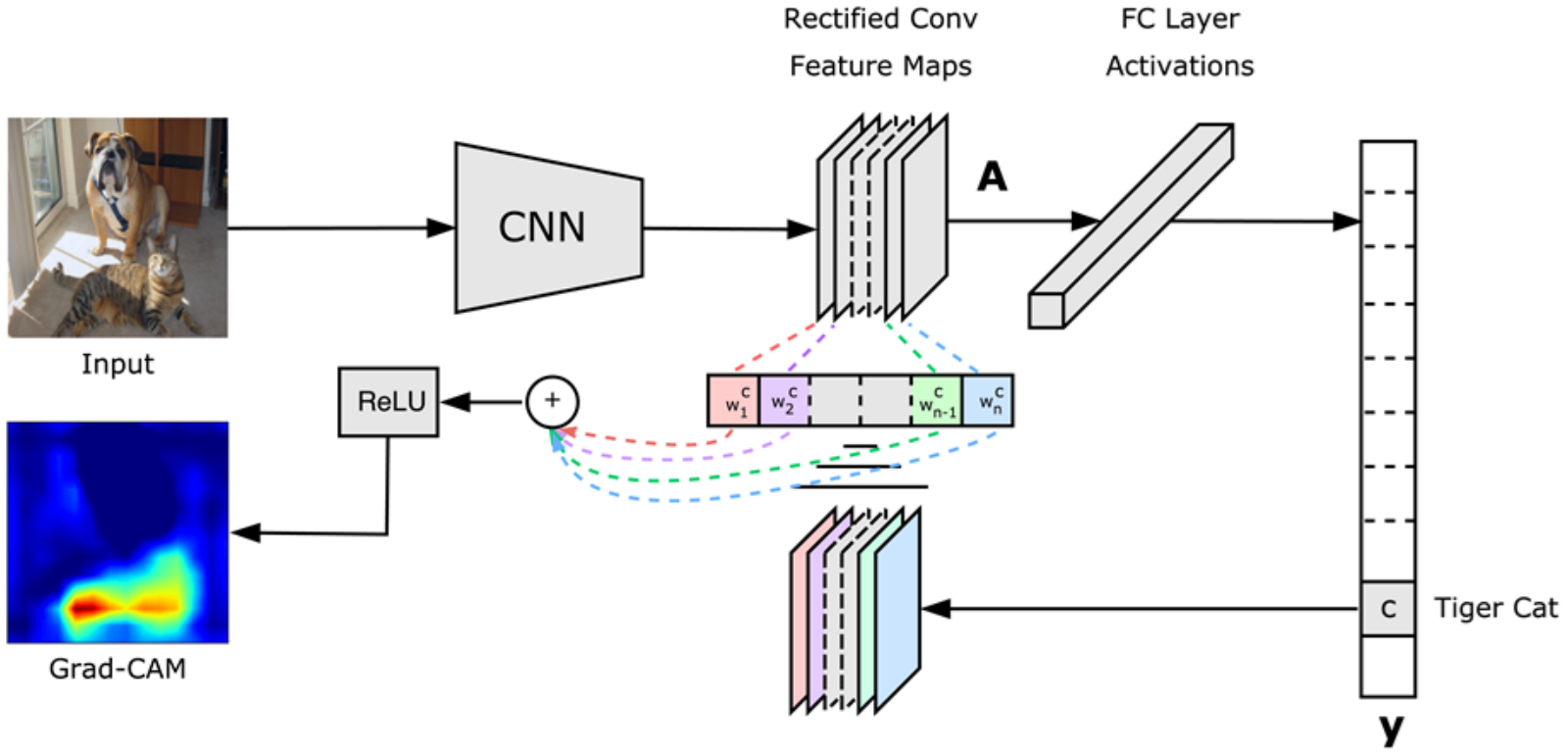
Zhou, Bolei, et al. "Learning deep features for discriminative localization." *CVPR*2016.

Grad-CAM

- برای هر لایه کانولوشنی، وزن هر نقشه فعالیت را بر اساس گرادیان خروجی کلاس مورد نظر نسبت به آن محاسبه می‌کند

$$w_k^c = \frac{1}{Z}\sum_i\sum_j \frac{\partial y^c}{\partial A_{ij}^k}$$

$$L^c_{\text{Grad-CAM}} = ReLU\left(\sum_k w_k^c\, A^k\right)$$



Selvaraju, R. R., et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." *ICCV2017*.

Listing 5.40 Loading the VGG16 network with pretrained weights

```
from keras.applications.vgg16 import VGG16

model = VGG16(weights='imagenet')
```

Note that you include the densely connected classifier on top; in all previous cases, you discarded it.

Listing 5.41 Preprocessing an input image for VGG16

```
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input, decode_predictions
import numpy as np

img_path = '/Users/fchollet/Downloads/creative_commons_elephant.jpg'

img = image.load_img(img_path, target_size=(224, 224))

x = image.img_to_array(img)

x = np.expand_dims(x, axis=0)

x = preprocess_input(x)
```

Local path to the target image

Python Imaging Library (PIL) image of size 224 × 224

float32 Numpy array of shape (224, 224, 3)

Adds a dimension to transform the array into a batch of size (1, 224, 224, 3)

Preprocesses the batch (this does channel-wise color normalization)

```
>>> preds = model.predict(x)
>>> print('Predicted:', decode_predictions(preds, top=3)[0])
```

```
>>> preds = model.predict(x)
>>> print('Predicted:', decode_predictions(preds, top=3)[0])

Predicted:', [(u'n02504458', u'African_elephant', 0.92546833),
(u'n01871265', u'tusker', 0.070257246),
(u'n02504013', u'Indian_elephant', 0.0042589349)]

>>> np.argmax(preds[0])

386
```

Listing 5.42 Setting up the Grad-CAM algorithm

"African elephant" entry in the prediction vector

```
african_elephant_output = model.output[:, 386]

last_conv_layer = model.get_layer('block5_conv3')
```

Output feature map of the block5_conv3 layer, the last convolutional layer in VGG16

Gradient of the "African elephant" class with regard to the output feature map of block5_conv3

Vector of shape (512,), where each entry is the mean intensity of the gradient over a specific feature-map channel

```
grads = K.gradients(african_elephant_output, last_conv_layer.output)[0]

pooled_grads = K.mean(grads, axis=(0, 1, 2))

iterate = K.function([model.input],
                     [pooled_grads, last_conv_layer.output[0]])

pooled_grads_value, conv_layer_output_value = iterate([x])

for i in range(512):
    conv_layer_output_value[:, :, i] *= pooled_grads_value[i]

heatmap = np.mean(conv_layer_output_value, axis=-1)
```

Values of these two quantities, as Numpy arrays, given the sample image of two elephants

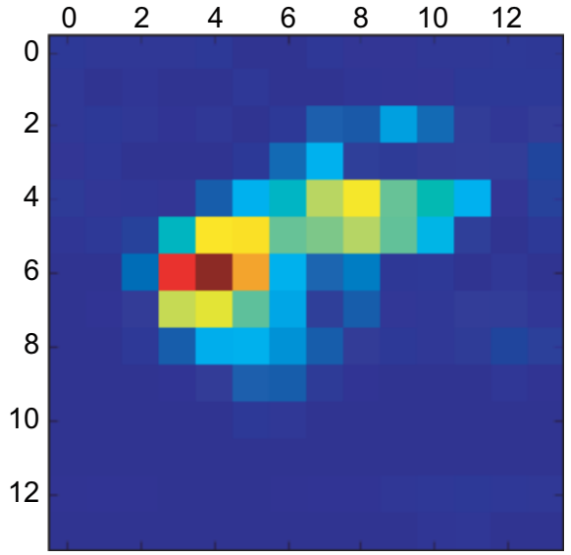
The channel-wise mean of the resulting feature map is the heatmap of the class activation.

Multiplies each channel in the feature-map array by "how important this channel is" with regard to the "elephant" class

Lets you access the values of the quantities you just defined: pooled_grads and the output feature map of block5_conv3, given a sample image

Listing 5.43 Heatmap post-processing

```
heatmap = np.maximum(heatmap, 0)
heatmap /= np.max(heatmap)
plt.matshow(heatmap)
```



Listing 5.44 Superimposing the heatmap with the original picture

```
import cv2
img = cv2.imread(img_path)
heatmap = cv2.resize(heatmap, (img.shape[1], img.shape[0]))
heatmap = np.uint8(255 * heatmap)
heatmap = cv2.applyColorMap(heatmap, cv2.COLORMAP_JET)
superimposed_img = heatmap * 0.4 + img
cv2.imwrite('/Users/fchollet/Downloads/elephant_cam.jpg', superimposed_img)
```

Uses cv2 to load the original image

Resizes the heatmap to be the same size as the original image

Converts the heatmap to RGB

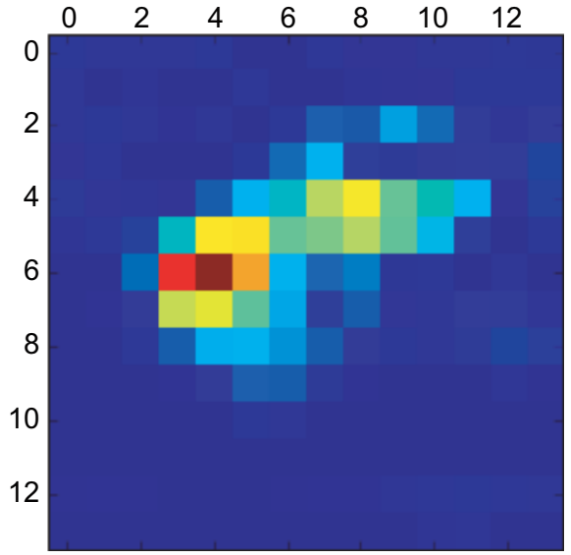
0.4 here is a heatmap intensity factor.

Applies the heatmap to the original image

Saves the image to disk

Listing 5.43 Heatmap post-processing

```
heatmap = np.maximum(heatmap, 0)
heatmap /= np.max(heatmap)
plt.matshow(heatmap)
```



نمایش نقشه‌های حرارتی (heatmaps)

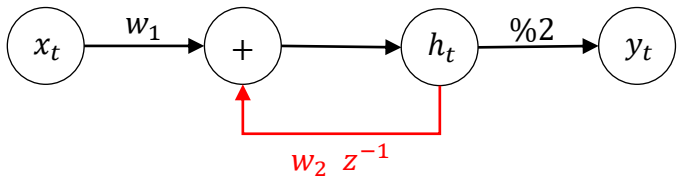
- این تکنیک نمایش به دو سوال مهم پاسخ می دهد:
  - چرا این شبکه تصمیم گرفت که این تصویر حاوی یک فیل آفریقایی است؟
  - فیل آفریقایی در کجای تصویر قرار دارد؟
- در این مثال، گوش‌های بچه فیل به شدت فعال شده‌اند
  - احتمالاً به این دلیل که این شبکه اینگونه می‌تواند تفاوت بین فیل‌های آفریقایی و هندی را تشخیص دهد

شبکه‌های عصبی بازگشتی

Recurrent Neural Networks

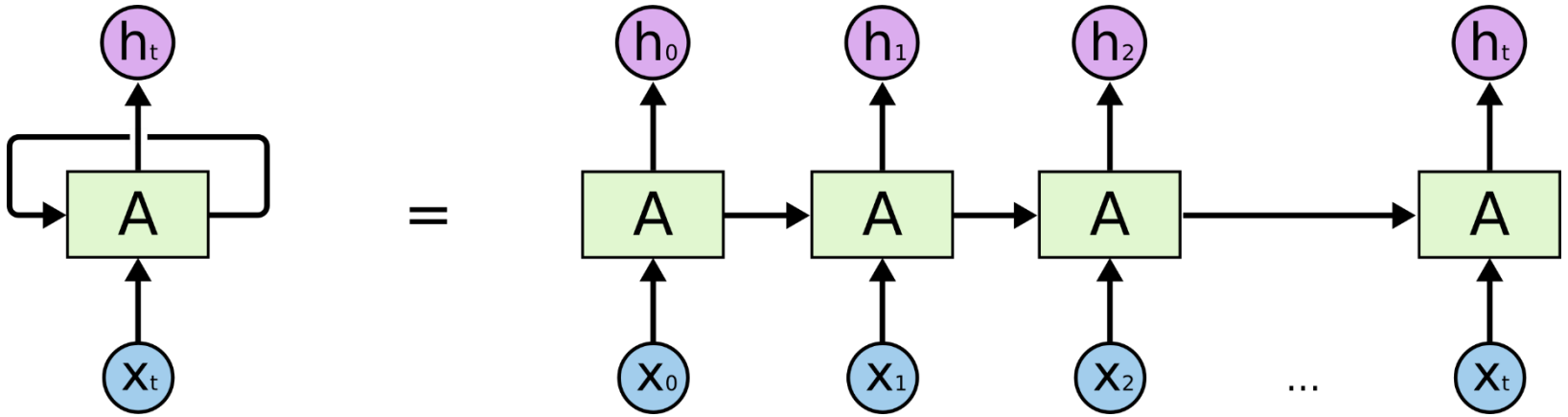
انگیزه

- در تمام مسئله‌ها نمی‌توان طول ورودی‌ها و خروجی‌ها را ثابت در نظر نگرفت
- در مسئله‌هایی مانند بازشناسی گفتار یا پیش‌بینی سری زمانی نیاز به سیستمی است که اطلاعات زمینه را ذخیره کند و از آنها به درستی استفاده نماید
- مثال ساده: اگر تعداد 1های یک دنباله فرد باشد خروجی 1 و در غیر اینصورت خروجی 0 تولید شود
  - 1000010101: 0 ، 1000110000000000000: 1 و ...
- انتخاب یک پنجره با طول ثابت سخت/غیرممکن است

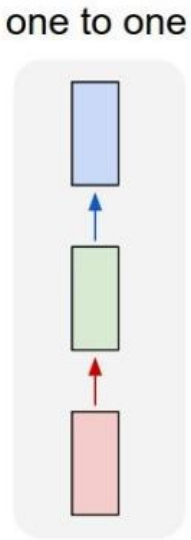


شبکه‌های عصبی بازگشتی

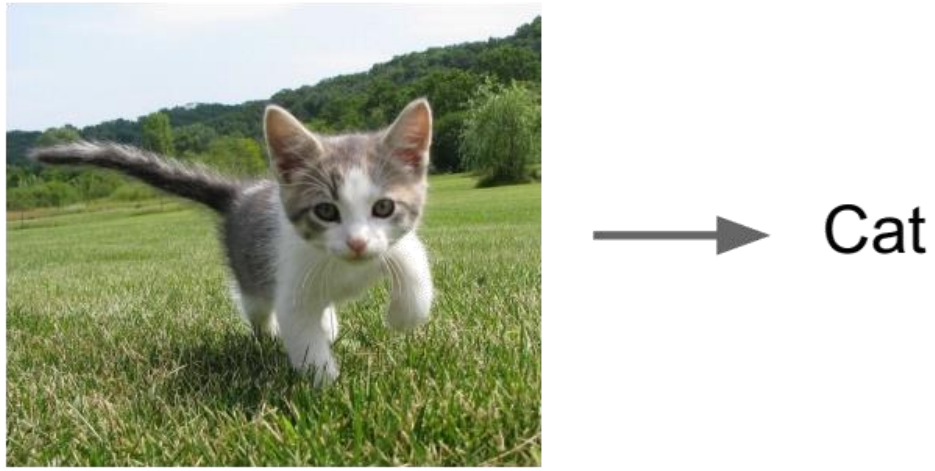
- شبکه‌های عصبی بازگشتی (RNNها) خانواده‌ای از شبکه‌های عصبی برای پردازش داده‌های دنباله‌ای هستند
  - دنباله‌ای از مقادیر $\boldsymbol{x}^{(1)}$، ...، $\boldsymbol{x}^{(\tau)}$
- اغلب شبکه‌های بازگشتی می‌توانند دنباله‌هایی با طول متغیر را نیز پردازش کنند
- یک RNN وزن‌های یکسانی را در چندین مرحله زمانی به اشتراک می‌گذارد



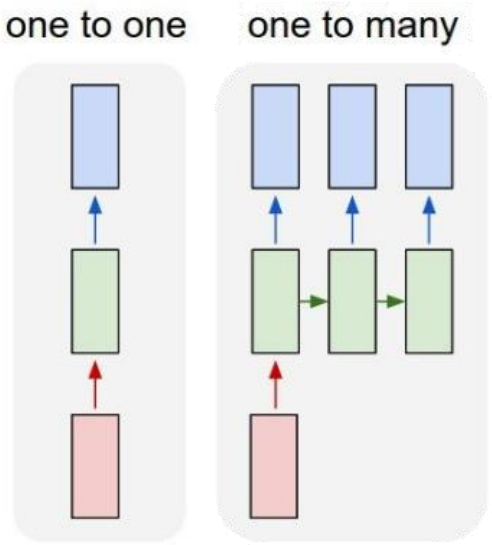
شبکه‌های عصبی



- لایه‌های Dense و Conv دارای حافظه نیستند!
  - ورودی‌های خود را به صورت مستقل پردازش می‌کنند (بدون هیچ حالتی در بین آنها)
- به چنین شبکه‌هایی پیش‌خور (feedforward) می‌گویند



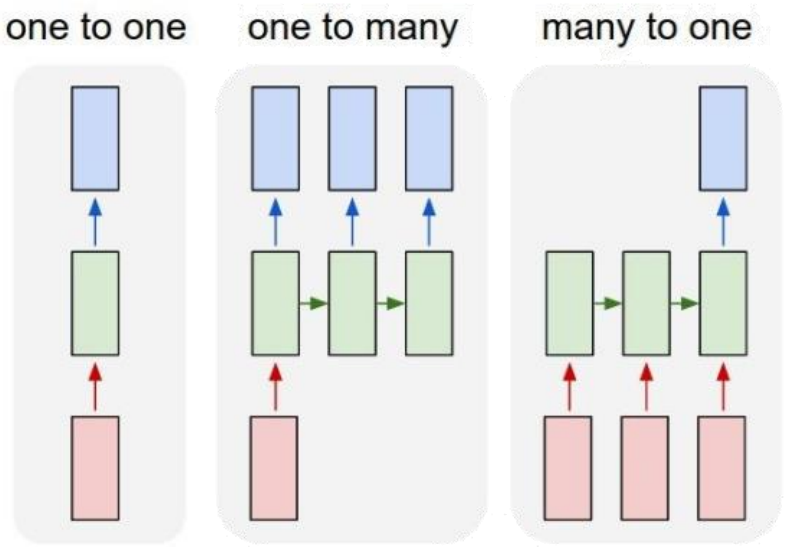
شبکه‌های عصبی



- خروجی این مثال دنباله‌ای از کلمات است که می‌تواند طول متغیر داشته باشد

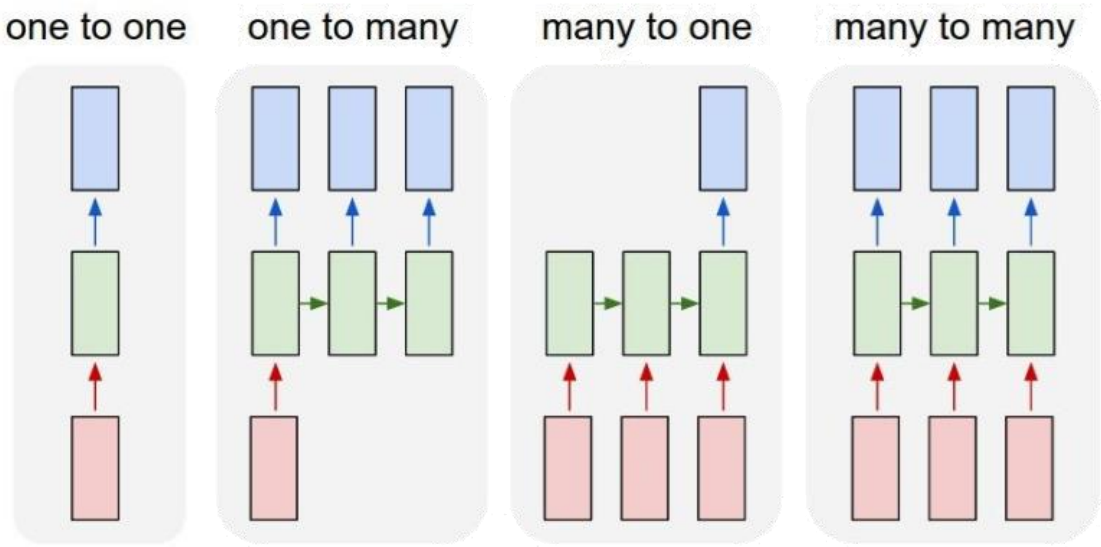
A cat is sitting on a tree branch

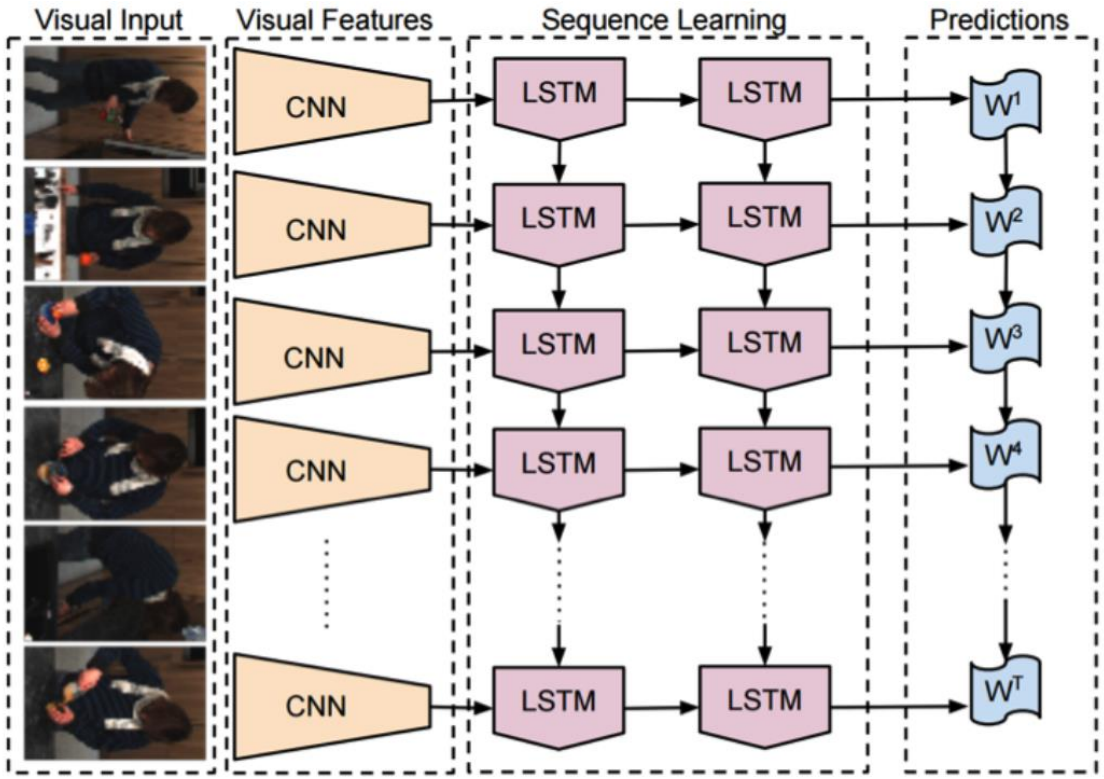
شبکه‌های عصبی



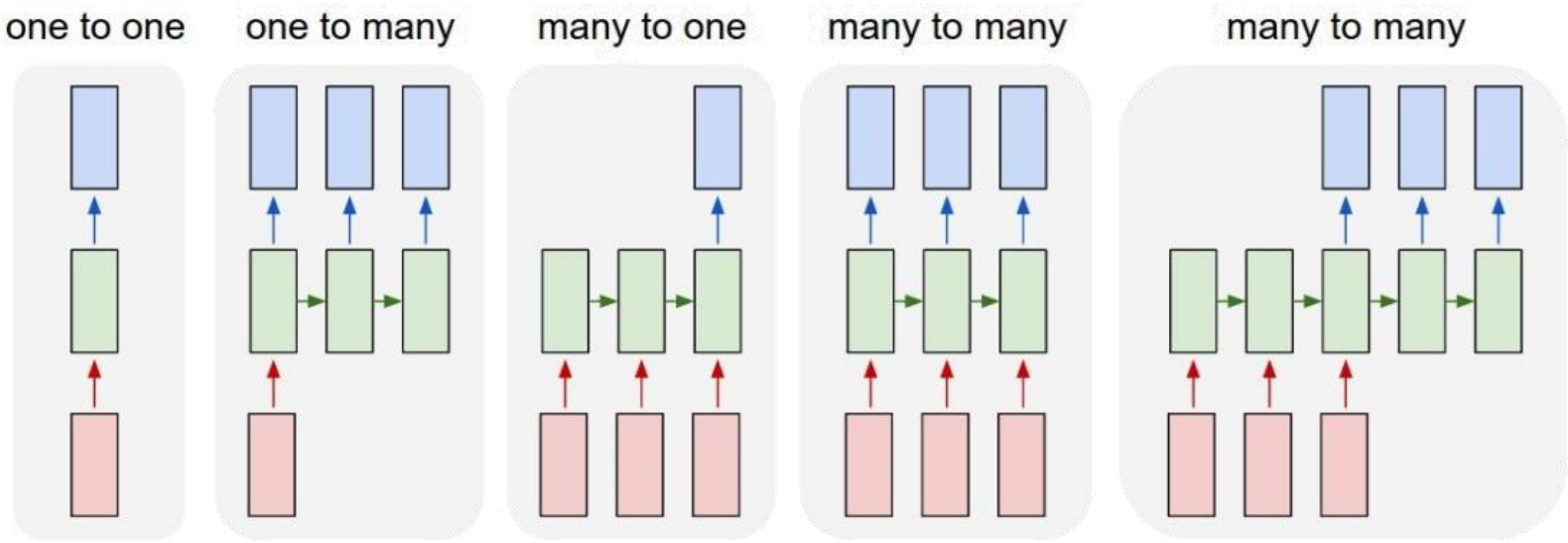
- در دسته‌بندی یک فایل صوتی، طول ورودی می‌تواند متغیر باشد

شبکه‌های عصبی





شبکه‌های عصبی



Input:	If you face a problem try to find the solution not the reason
Output:	مشکل که به وجود اومد بگرد راه حلش را پیدا کن نگرد دنبال این که چرا به وجود اومد