# Introdução a Aprendizagem “profunda”

*deep learning*

*deep neural networks*

Prof.: Eric A. Antonelo

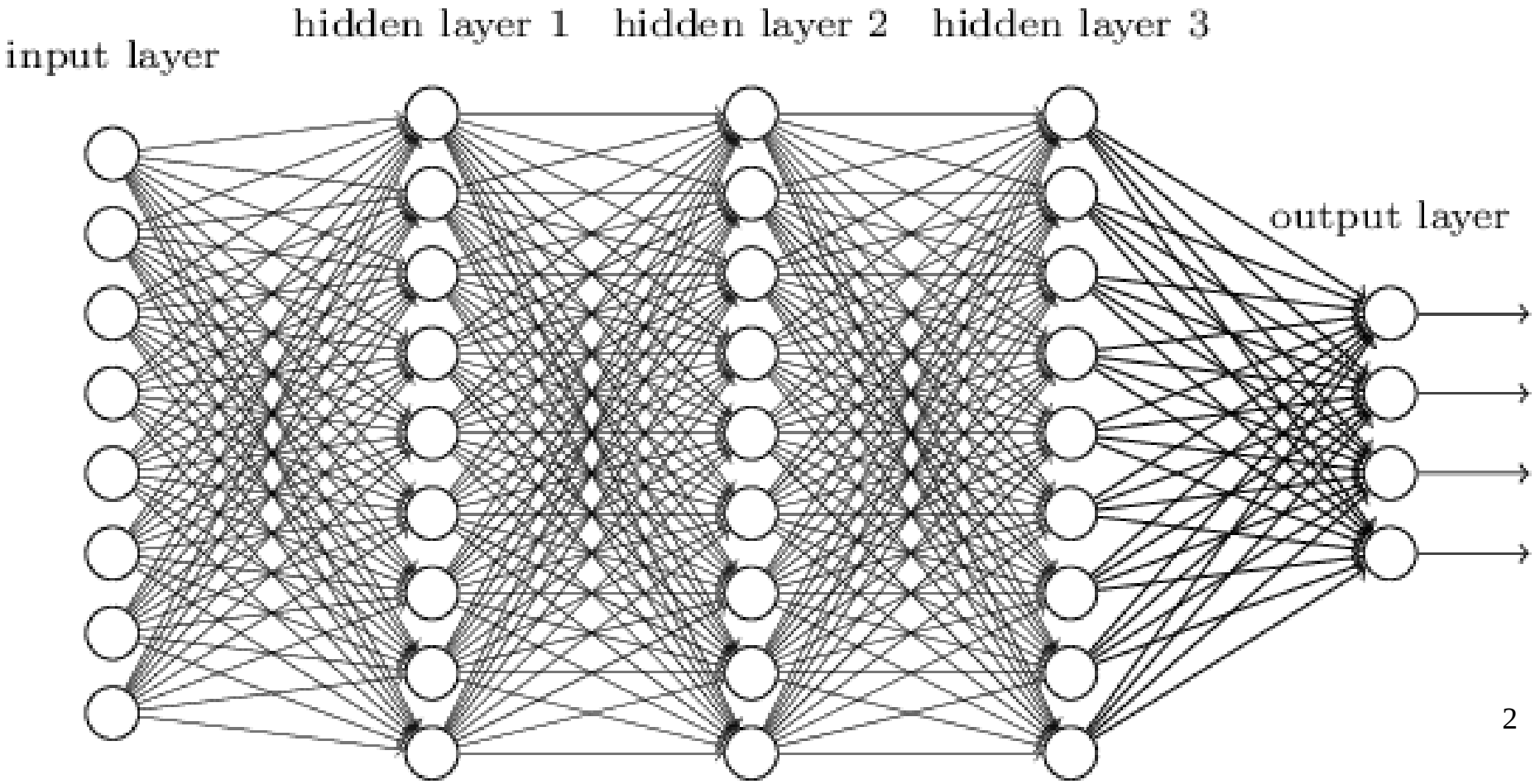
Disciplina: Inteligência Artificial

DAS-CTC Florianópolis

Maio/2015

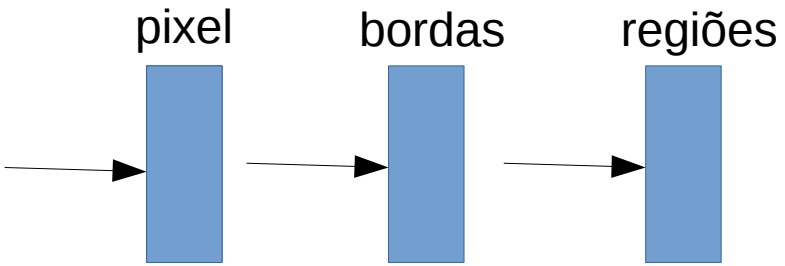
# O que são Redes Neurais Profundas?

- São redes com múltiplas camadas intermediária ou ocultas (>= 2)



# O que são Redes Neurais Profundas?

- São redes com múltiplas camadas intermediárias ou ocultas (>= 2)
- A cada camada, **conceitos mais abstratos** podem ser aprendidos a partir de conceitos menos abstratos (da camada anterior)
- Assim, cada camada aprende uma representação dos dados em um certo nível de abstração:
  - Uma imagem pode ser representada como um conjunto de bordas, regiões de contorno particular, etc.



# Aplicações Comuns

- Muito comum em
  - classificação de imagens (visão computacional)
  - reconhecimento automático de voz,
  - processamento de linguagem natural
  - reconhecimento de áudio
  - e bioinformática

  contendo uma grande quantidade de exemplos (**Larga escala**)

  E alcançando resultados no estado-da-arte

# Tipos

- Muitas vezes, grande parte dos exemplos não estão categorizados:
  - Aprendizagem não supervisionada
  - Aprendizagem semi-supervisionada de features
- Algumas representações são inspiradas por avanços na neurociência

TIPOS:

- Redes neurais profundas (*deep neural networks*)
- **Redes neurais profundas com convolução**
  ***(CNN – convolutional neural networks)***
- Redes profundas de crença (*deep belief networks*)

# Redes neurais profundas com convolução (CNN)

- RNA onde neurônios individuais são posicionados de modo que eles respondam a **regiões sobrepostas do campo visual**
- Inspirados por:
  - Processos biológicos
  - Variações de perceptron multicamadas

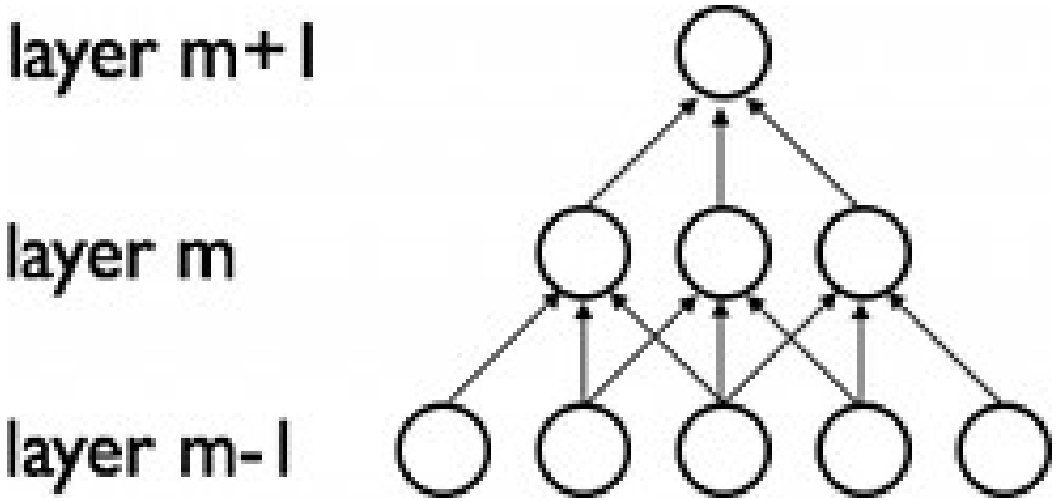
  para usar **quantidade mínimas de pré-processamento**
- Muito usadas no reconhecimento de imagens e video.

# Redes neurais profundas com convolução (CNN)

- **Campos receptivos**:
  - um neurônio recebe sinais somente de uma *pequena porção da entrada*
  - Há sobreposição de um campo receptivo com outro
  - Toleram translação de uma imagem de entrada

**Conectividade esparsa**

**Neurônios atuam como filtros locais sobre o espaço de entradas**

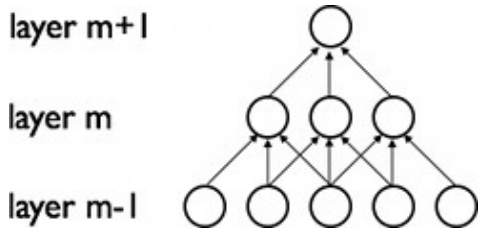


# Redes neurais profundas com convolução (CNN)

- Resultados da neurociência (córtex visual):

  2 tipos de células:

  - Células simples: respondem ao máximo para padrões específicos do tipo borda no seu campo receptivo
  - Células complexas: tem campos receptivos maiores e são invariantes localmente à posição exata do padrão



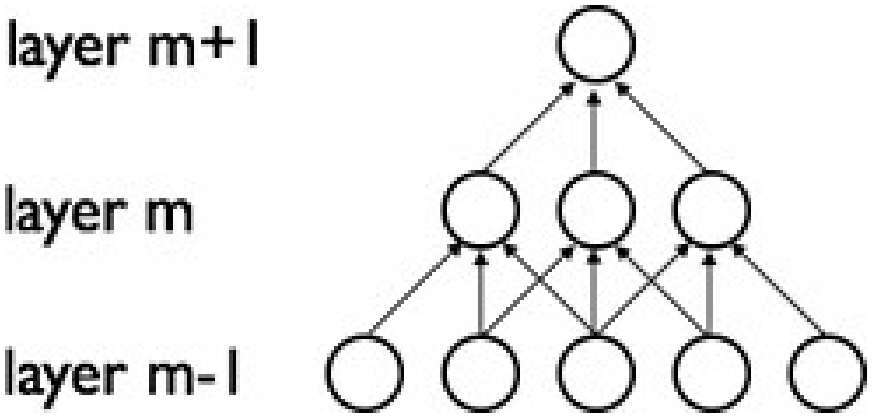
# Conectividade esparsa (CNN)

- Padrão de **conectividade local**

  r

  - As entradas da camada **m** advem de um sub-conjunto de unidades da camada **m-1**

- **m-1**: retina
- **m**: tem campos receptivos de largura 3
- **m+1**: conectividade igual; mas com relação a **m-1**, tem um campo receptivo de largura 5

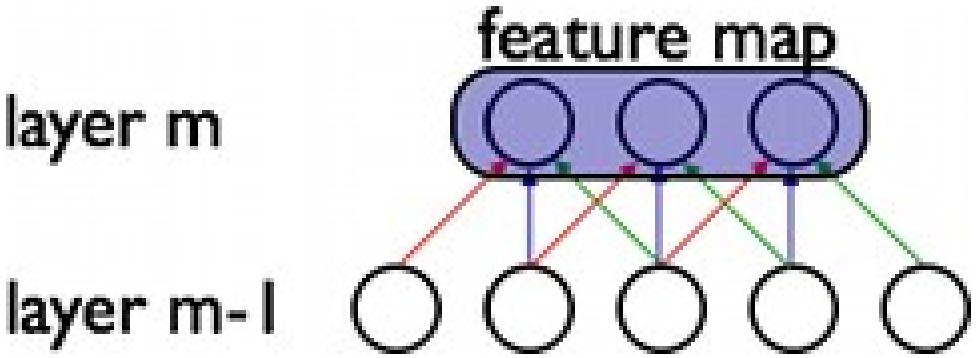



Empilhando-se várias camadas resulta em “filtros” (não-lineares) que se tornam **gradativamente mais globais**.

# Pesos compartilhados (CNN)

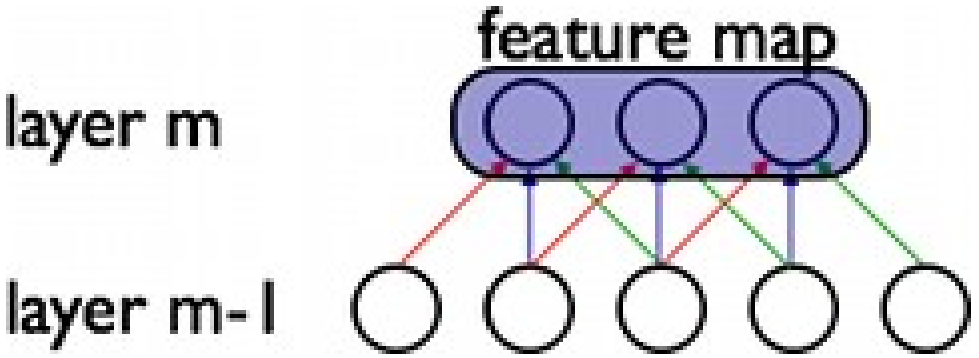
- Em CNNs, cada filtro $h_i$ é replicado através de todo campo visual
- As unidades **compartilham a mesma parametrização** (vetor de pesos e bias), formando um mapa de *features*.

3 unidades ocultas compartilham os mesmos pesos: pertecem ao mesmo mapa de features



# Pesos compartilhados (CNN)

- Vantagens:
  - Reduz o número de parâmetros livres para ser aprendido
  - Tais restrições melhoram a capacidade de generalização em problemas de visão computacional.



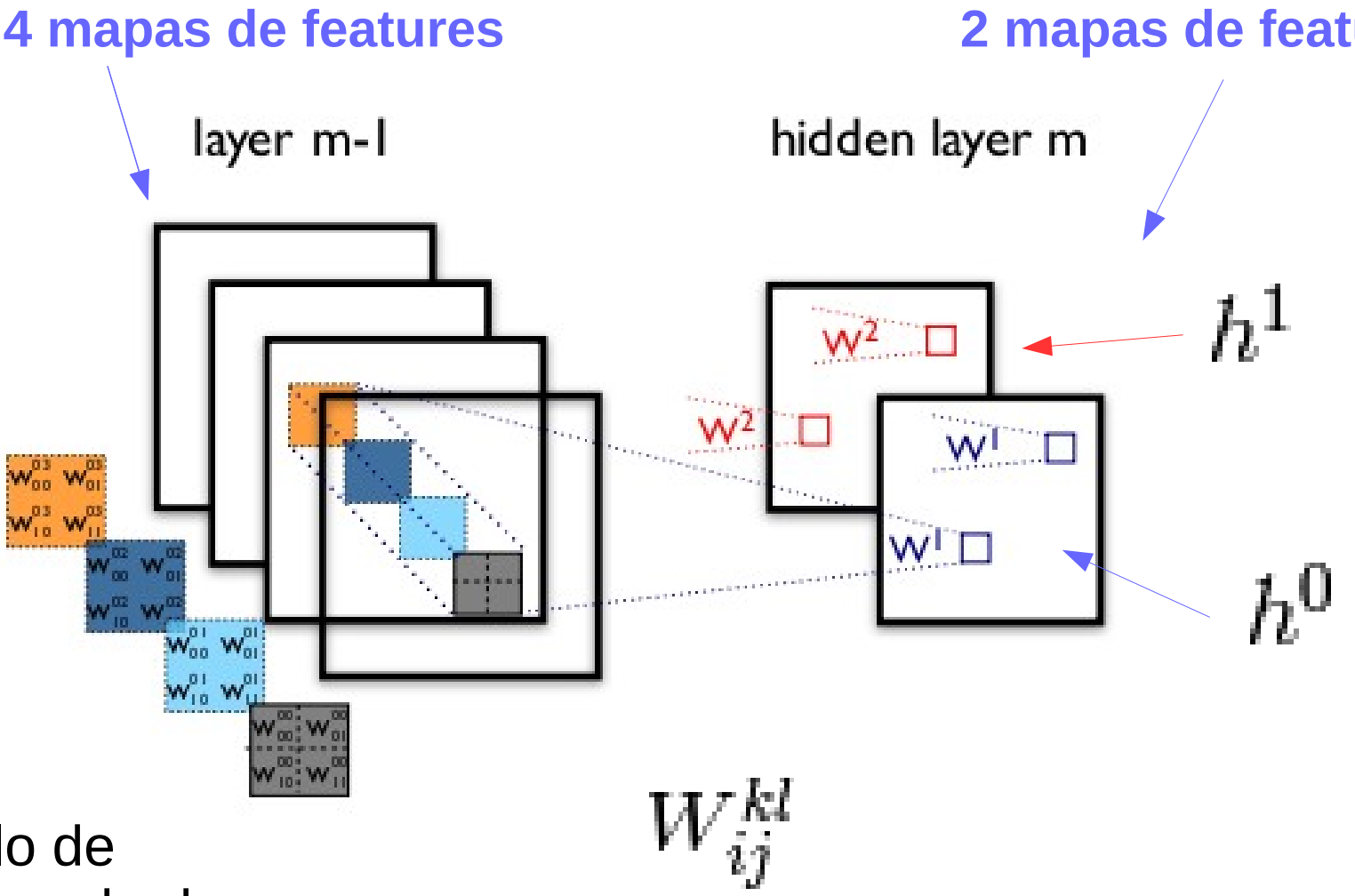
# Detalhes e Notação

Um mapa de features é obtido pela aplicação repetida de uma função através da imagem inteira. Isto é, pela convolução da imagem de entrada com um filtro linear, adição de um termo bias e então aplicação de uma função linear.

$$h_{ij}^{k} = \tanh((W^{k} * x)_{ij} + b_{k}).$$

Para obter uma representação mais rica dos dados, utilizam-se múltiplos mapas de features por camada.

# Detalhes e Notação




Exemplo de uma camada de convolução

O peso conectando a unidade **k** da camada **m**, com a unidade em **(i,j)** do **l-esimo** mapa de features da camada **m-1**

# O operador de convolução em Theano

**Python library**

- theano.tensor.signal.conv2d

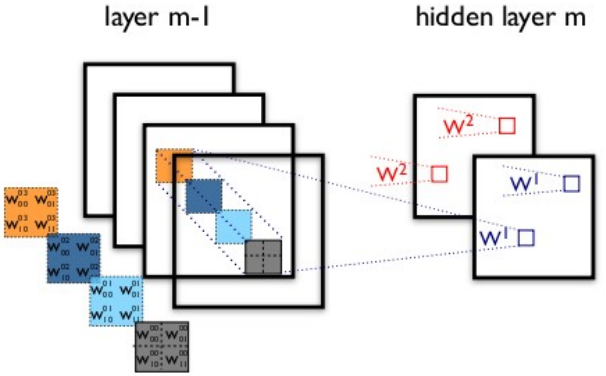
  2 entradas simbólicas :

  - Um tensor 4D correspondendo a um mini-batch de imagens de entrada. O formato é: *[mini-batch size, number of input feature maps, image height, image width].*
  - Um tensor 4D correspondendo à matriz de pesos **W**. O formato é: *[number of feature maps at layer m, number of feature maps at layer m-1, filter height, filter width].*

# O operador de convolução em Theano

**Python library**

- Exemplo código ipython
  - Entrada consiste de 3 mapas de entrada (RGB) de tamanho por ex. 120x160 ou algum outro.
  - 2 filtros de convolução com campos receptivos de 9x9.




*[number of feature maps at layer m, number of feature maps at layer m-1, filter height, filter width].*

```
w_shp = (2, 3, 9, 9)
```

Pesos inicializados aleatoriamente no intervalo:
**[-1/fan-in, 1/fan-in]**

**Fan-in**: número de entradas para uma unidade oculta.

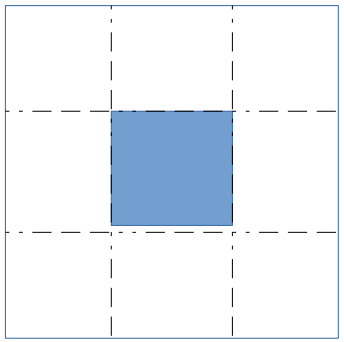
# MaxPooling

- Uma forma de amostragem não-linear (*nonlinear downsampling*)
- MaxPooling
  - particiona a imagem de entrada em um conjunto de retângulos contíguos e não sobrepostos,
  - e para cada retângulo gera uma saída igual ao máximo valor encontrado.

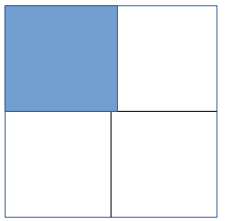
# MaxPooling

- Útil porque:
  - Eliminando valores não-máximos, reduz computação para camadas posteriores.
  - Provê uma forma de invariância à translação

Possíveis direções:



Max-pooling 2x2:



3 de 8 configurações possíveis resultam na mesma saída

Max-pooling 3x3:  5 de 8 resulta na mesma saida

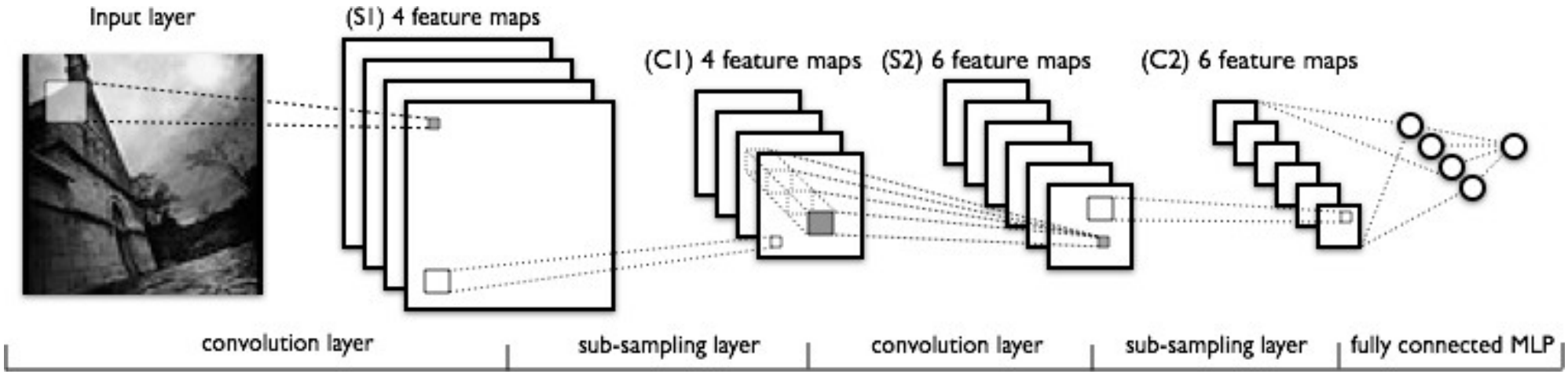
# MaxPooling em Theano

- theano.tensor.signal.downsample.max_pool_2d
  - entrada: tensor de dimensão n, (n>=2), e um fator de *downscaling*

- Exemplo ipython

# O modelo completo: LeNet

- Camadas inferiores alternam-se entre camadas de convolução e max-pooling.
- Camadas superiores correspondem a um MLP (rede multicamadas + regressão logística)



# O modelo completo: LeNet

- Executar Código no banco de dados MNIST

# Referências

http://deeplearning.net/tutorial/lenet.html