Reconhecedor multivariado

# Reconhecedor de caracteres escritos manualmente

...

Luiz Eduardo Sol - 8586861
Henrique Melo - 9347031

https://docs.google.com/presentation/d/1BZ28XVNu81dZnDUGawMGHzehjWO6nqSvTUxQSeU8aJE/edit?usp=sharing

PSI3571 - Práticas em Reconhecimento de Padrões, Modelagem e Inteligência Computacional

**Objetivo do projeto:**
Elaborar um reconhecedor de caracteres escritos manualmente em uma imagem

# Variáveis

| Entrada | Saída |
| --- | --- |
| Imagem X x X pixels em níveis de cinza | “Probabilidade” de cada caracter |

# Etapas do projeto

| | |
|---|---|
| Coleta e tratamento do dados | • Obter e tratar conjuntos de treino / teste / validação |
| Estudo e escolha do método de modelagem | • Estudo de possíveis modelos (inclusive soluções já propostas) |
| Desenvolvimento e aprimoramento do modelo | • Definição dos melhores parâmetros / técnicas para o modelo escolhido |
| Compilação de resultados | • Compilação de métricas de desempenho interessantes e apresentação de resultados |

# Ferramentas utilizadas

- Linux
- GitHub
- Jupyter
- Python 3
  - NumPy
  - Keras

# Coleta e tratamento dos dados

- Banco de dados EMNIST obtido facilmente por meio de uma biblioteca desenvolvida para python (https://pypi.org/project/python-mnist/)

```
pip install python-mnist
```

# O dataset EMNIST

- Disponibilizado pelo National Institute of Standards and Technology (EUA) (https://www.nist.gov/)
- É um dataset de caracteres escritos manualmente em imagens 28 x 28 pixels
- Disponibilizado no formato MATLAB (.mat) ou em arquivos binários
-

Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. Retrieved from http://arxiv.org/abs/1702.05373

# Modelo desenvolvido

- Baseado em solução proposta disponível em: https://github.com/shubhammor0403/EMNIST/blob/master/modeltrain.ipynb
- Modelo por rede neural

# Detalhes do modelo: as camadas

- 1x Reshape
  - *model.add(Reshape((28, 28, 1), input_shape=(784, )))*
- 2x Convolução 2D com ativação relu
  - *model.add(Convolution2D(32, (5,5), input_shape=(28,28,1), activation='relu', ...))*
- 1x MaxPooling2D
  - *model.add(MaxPooling2D(pool_size=(2,2)))*
- 1x Flatten
  - *model.add(Flatten())*
- 2x Dense
  - *model.add(Dense(1024, activation='relu', kernel_constraint=maxnorm(3)))*
  - *model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))*
- 1x Dropout
  - *model.add(Dropout(0.5))*
- 1x Dense Softmax (saída)
  - *model.add(Dense(62, activation='softmax'))*

# Reshape

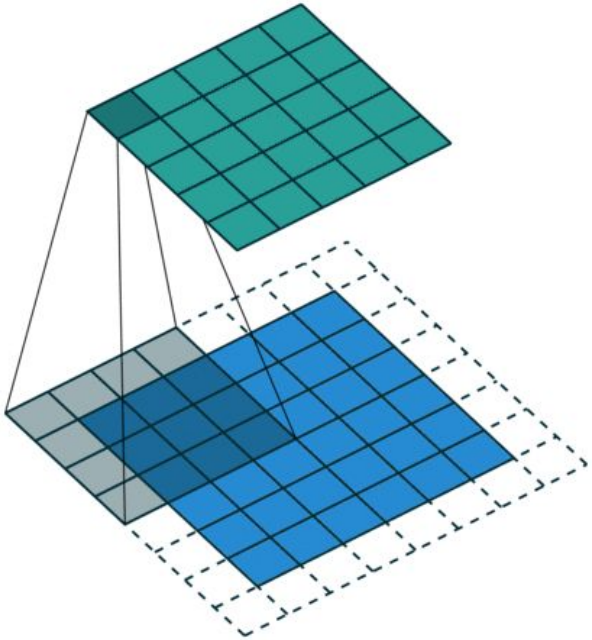
Transforma um vetor de entrada em uma matrix

[1, 2, 3, 4, 5, ...] -> [[1, 2, 3],

[4, 5, 6], ...]

# Convolução 2D

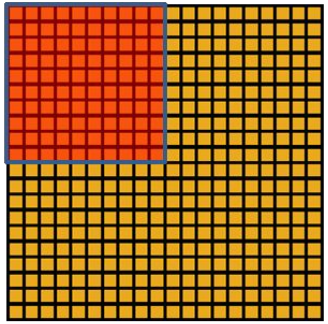
Aplica um filtro convolucional na matriz gerada pelas camadas anteriores.



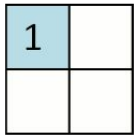
Fonte: http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

# Maxpooling 2D

Semelhante ao filtro convolucional, mas captura o valor máximo da região do *kernel*.



Convolved feature




Pooled feature

Fonte:
https://hackernoon.com/visualizing-parts-of-convolutional-neural-networks-using-keras-and-cats-5cc01b214e59

# Flatten

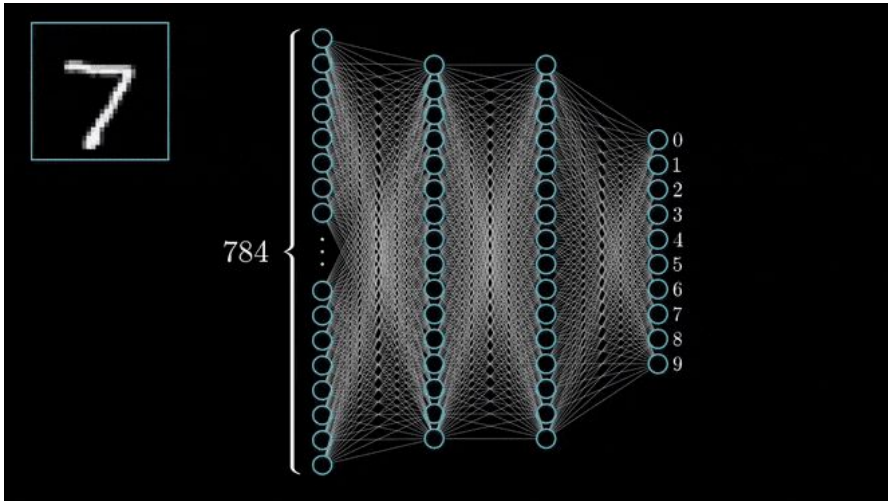
Transforma matrizes em vetores.

[[1, 2, 3],

[4, 5, 6], ...] -> [1, 2, 3, 4, 5, ...]

# Dense

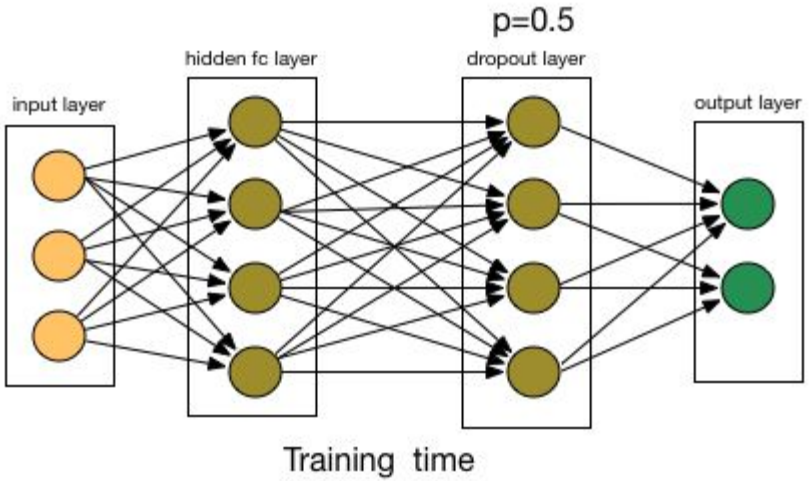
Camada de *perceptrons* completamente conectados às camadas anteriores e posteriores (*a.k.a. fully connected*).




Fonte: https://gfycat.com/gifs/search/neural+network

# Dropout

Desativa aleatoriamente alguns *perceptrons* durante o treinamento para evitar *overfitting*.




Fonte: http://primo.ai/index.php?title=Dropout

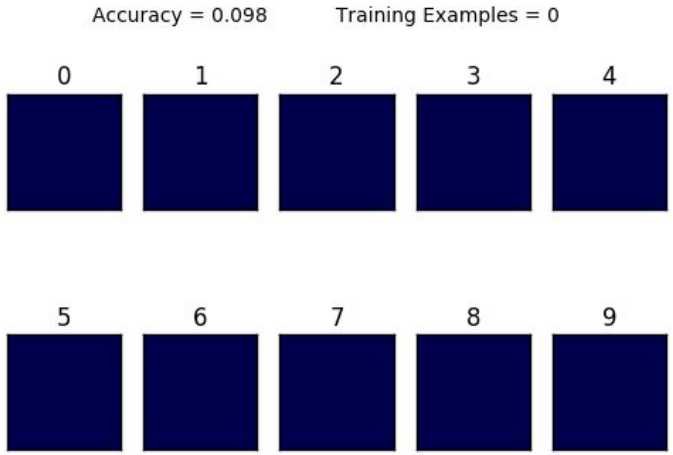
# Softmax

Atribui à saída de maior valor o valor 1 e a todas outras 0. É soft por ser uma função max diferenciável (necessário para *backpropagation*).

| | | |
|---|---|---|
| [0.34, | | [0, |
| 0.664, | | 0, |
| 0.00123, | -> | 0, |
| 0.90123] | | 1] |

# Treinamento

Utilizamos o algoritmo *AdamMax* e a *cross-entropy* como loss function por 20 épocas (6.5 h) no CPU com 12 threads @ 4GHz e 40GB de RAM.

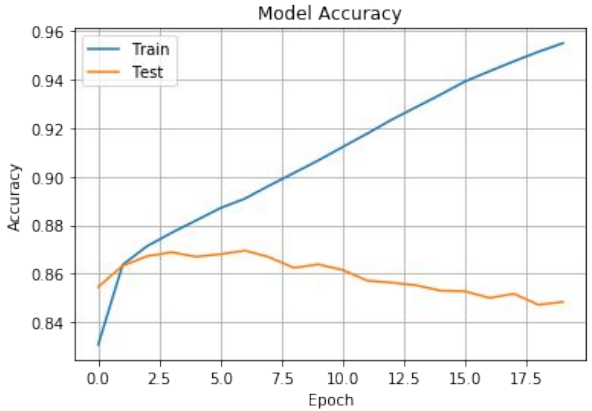


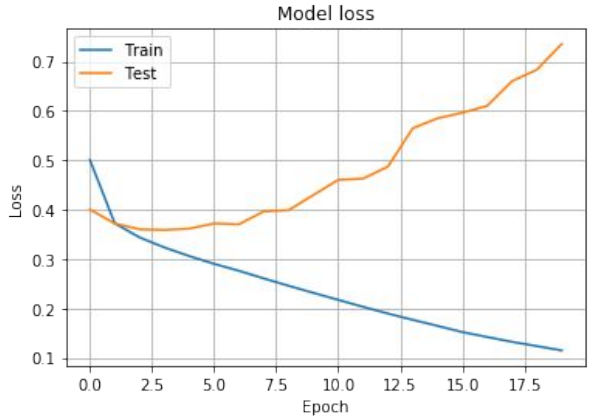

Fonte: https://www.oreilly.com/learning/not-another-mnist-tutorial-with-tensorflow

# Resultado

Acurácia: 84,89%

Acurácia do treino melhora enquanto a acurácia do teste diminui.



Todo código desenvolvido no projeto pode ser visualizado e obtido no repositório do GitHub: https://github.com/henriquemeloo/psi3571-handwriting-recognition

# Obrigado