# Modelo de uma Rede Neural Profunda para reconhecimento de texto manuscrito *off-line*

Aluno: Caíque Reinhold

Orientadora: Luciana Pereira de Araújo

FURB
UNIVERSIDADE DE BLUMENAU

# Roteiro

- Introdução
- Objetivos
- Fundamentação teórica
- Trabalhos correlatos
- Especificação
- Implementação
- Resultados e discussões
- Conclusões e sugestões

# Introdução

- Recentes avanços do aprendizado profundo
- Complexidade do reconhecimento de texto manuscrito
- Armazenamento digital de informações

# Objetivos

- Apresentar um modelo de rede neural para reconhecimento de texto manuscrito em imagens digitais
- Utilizar aprendizado profundo para reconhecimento de padrões visuais
- Desenvolver um protótipo para validação do modelo apresentado

# Fundamentação Teórica

- Reconhecimento de texto manuscrito:
  - De marcas gráficas para digital
  - *Online/off-line*
  - Enorme variabilidade dos padrões
- Aprendizado profundo:
  - Motivação
  - Dificuldade no treinamento
  - Recente expansão

# Fundamentação Teórica

- Redes Neurais:
  - Baseadas na estrutura do cérebro
  - Unidades de processamento similares a neurônios
  - Foco do trabalho:
    - *Multilayer Perceptron*
    - Redes Convolucionais
    - *Long Short-Term Memory*

# Fundamentação Teórica

- *Multilayer Perceptron*:
  - Aproximador universal
  - Densamente conectada

# Fundamentação Teórica

- Rede convolucional:
  - Baseada na estrutura do córtex visual
  - Conectividade esparsa
  - Pesos compartilhados
  - Amostragem
  - Recordes no reconhecimento de padrões visuais

# Fundamentação Teórica

- *Long Short-Term Memory*:
  - Rede recorrente
  - Célula de memória
  - Portões de ativação

# Fundamentação Teórica

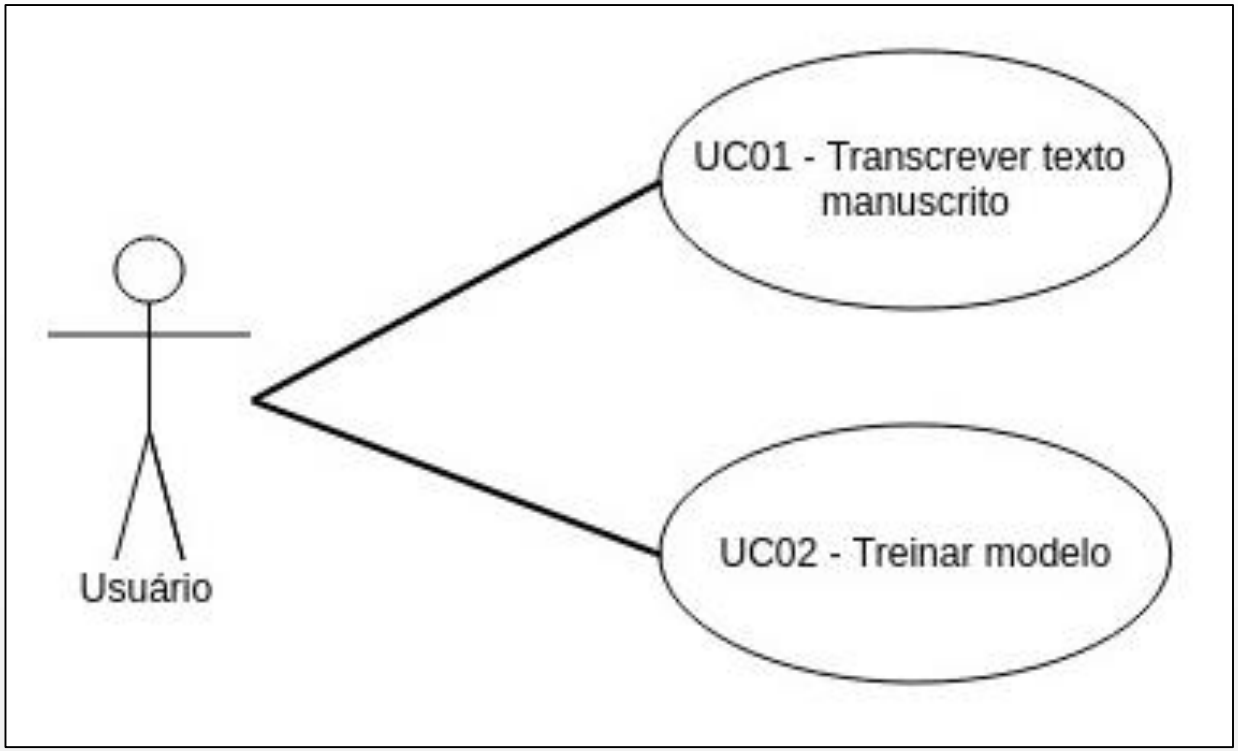
- *Connectionist Temporal Classification*:
  - Classificação temporal
  - Função de custo
  - Probabilidades dos alinhamentos

# Trabalhos Correlatos

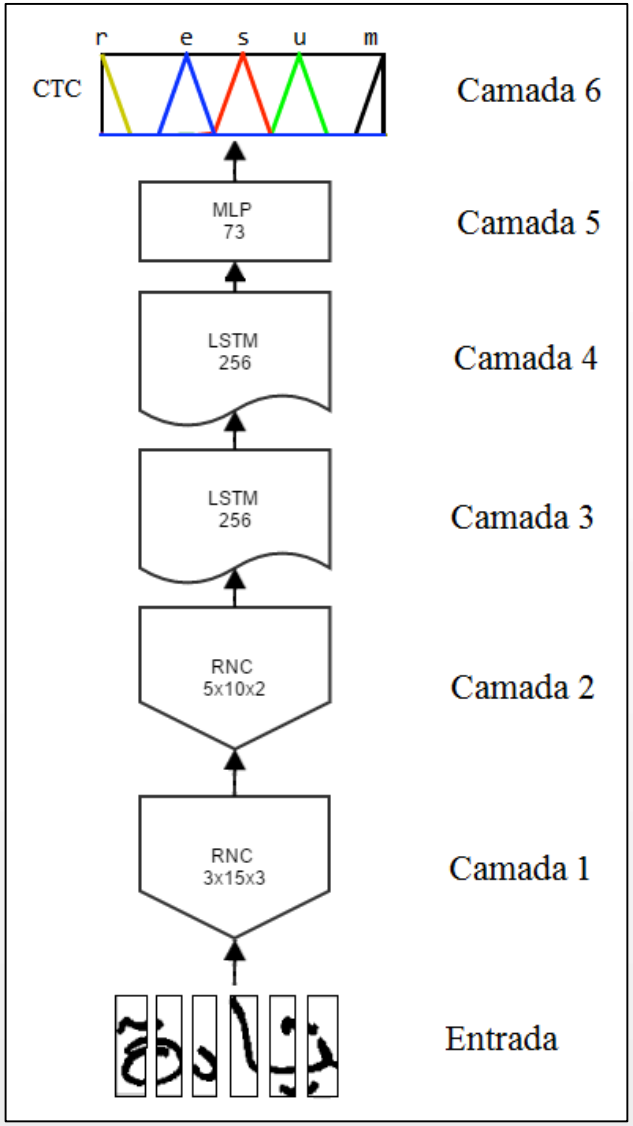
- Offline Arabic Handwriting Recognition with Multidimensional Recurrent Neural Networks (GRAVES, 2012)
- Segmented Handwritten Text Recognition with Recurrent Neural Network Classifiers (SU et al., 2015)
- A Context-Sensitive-Chunk BPTT Approach to Training Deep LSTM/BLSTM Recurrent Neural Networks for Offline Handwriting Recognition (CHEN; YAN; HUO, 2015)

# Especificação

Casos de uso



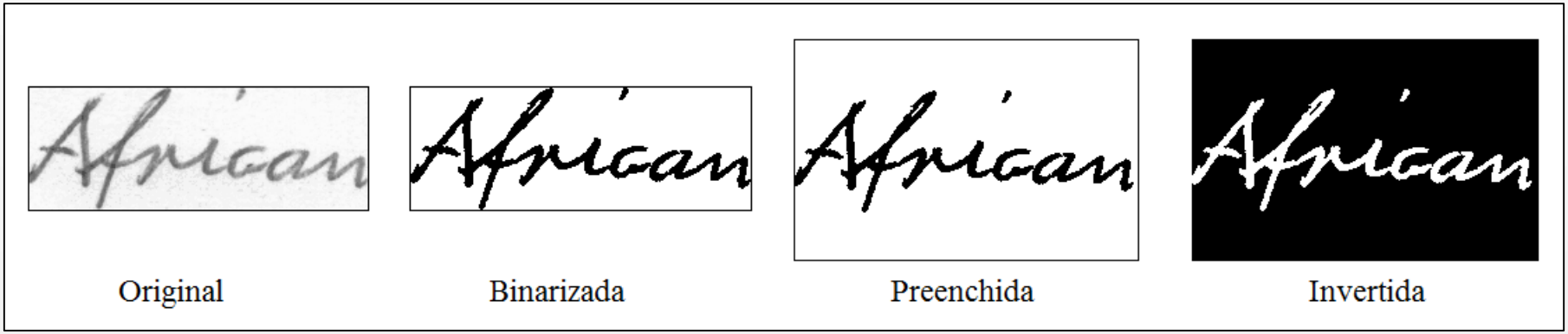
# Arquitetura do modelo



# Implementação

Pré-processamento:

- Binarização
- Padding
- Inversão



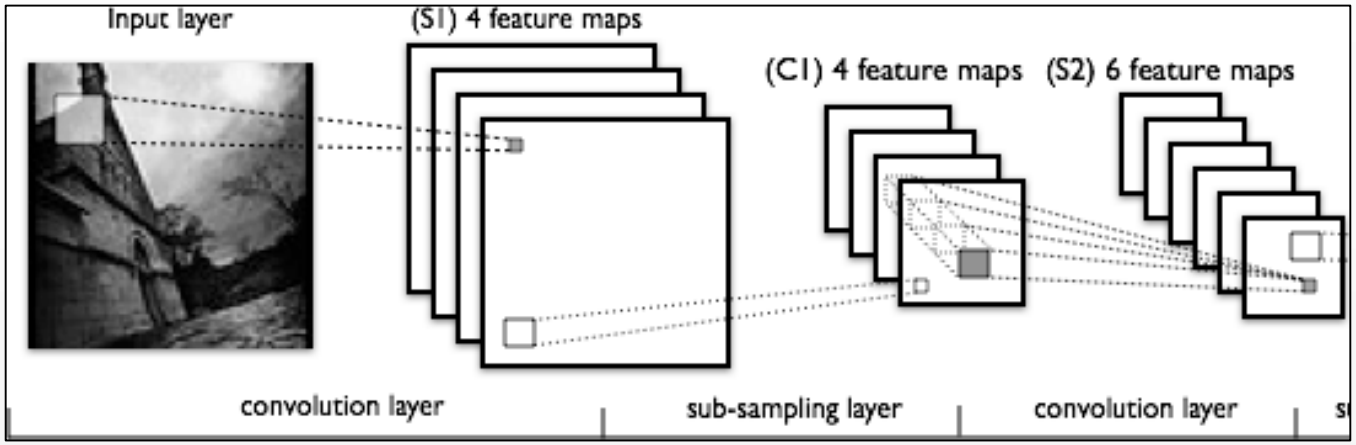
# Implementação

*Multilayer Perceptron*

```
lin_output = tt.dot(self.input, self.W) + self.b
self.output = (lin_output if activation == None
               else activation(lin_output))
```

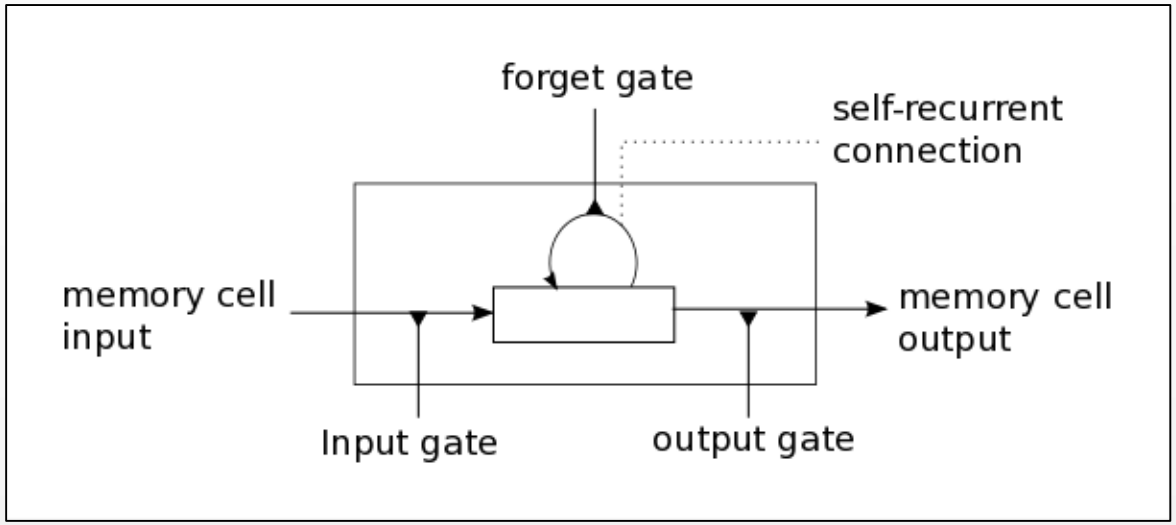
# Implementação

Convolucional



# Implementação

*Long Short-Term Memory*



# Implementação

*Connectionist Temporal Classification*

```
[[ 1.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  1.,  1.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  1.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  1.,  1.,  1.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  1.,  1.,  0.,  0.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  1.,  1.,  1.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  1.,  1.,  0.,  0.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  1.,  1.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  1.,  0.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  1.],
 [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.]]
```

```
probs, _ = theano.scan(
    lambda curr, accum: curr * tt.dot(accum, recurrence_relation),
    sequences=[pred_y],
    outputs_info=[tt.eye(self.y.shape[0])[0]]
)
```

# Implementação

Modelo

```
self.layer0 = ConvPoolLayer(
    filter_shape=(fn1, 1, fh1, fw1),
    image_shape=(1, 1, h, w),
    params=params[0] if params else None
)

h2, w2 = ((h - fh1 + 1) / 2), ((w - fw1 + 1) / 2)
self.layer1 = ConvPoolLayer(
    filter_shape=(fn2, fn1, fh2, fw2),
    image_shape=(1, fn1, h2, w2),
    params=params[1] if params else None
)

def conv(index, X):
    slice_x = X[:, index * w:(index + 1) * w]
    return self.layer1.output(self.layer0.output(
        slice_x.dimshuffle('x', 'x', 0, 1)
    )).flatten(1)
    # return slice_x.flatten(1)

conv_out, _ = theano.scan(
    fn=conv, sequences=[tt.arange(self.input.shape[1] / w)],
    non_sequences=self.input, outputs_info=None
)
```

# Implementação

Modelo

```
self.layer2 = LSTMLayer(
    input=conv_out,
    in_size=((h2 - fh2 + 1) / 2) * ((w2 - fw2 + 1) / 2) * fn2,
    out_size=256,
    params=params[2] if params else None
)

self.layer4 = LSTMLayer(
    input=self.layer2.output,
    in_size=256,
    out_size=256,
    params=params[4] if params else None
)

self.layer3 = DenseLayer(
    input=self.layer4.output,
    in_size=256,
    out_size=len(CLASSES) + 1,
    activation=None,
    params=params[3] if params else None
)

self.ctc = CTCLayer(
    input=self.layer3.output,
    y=self.y
)
```

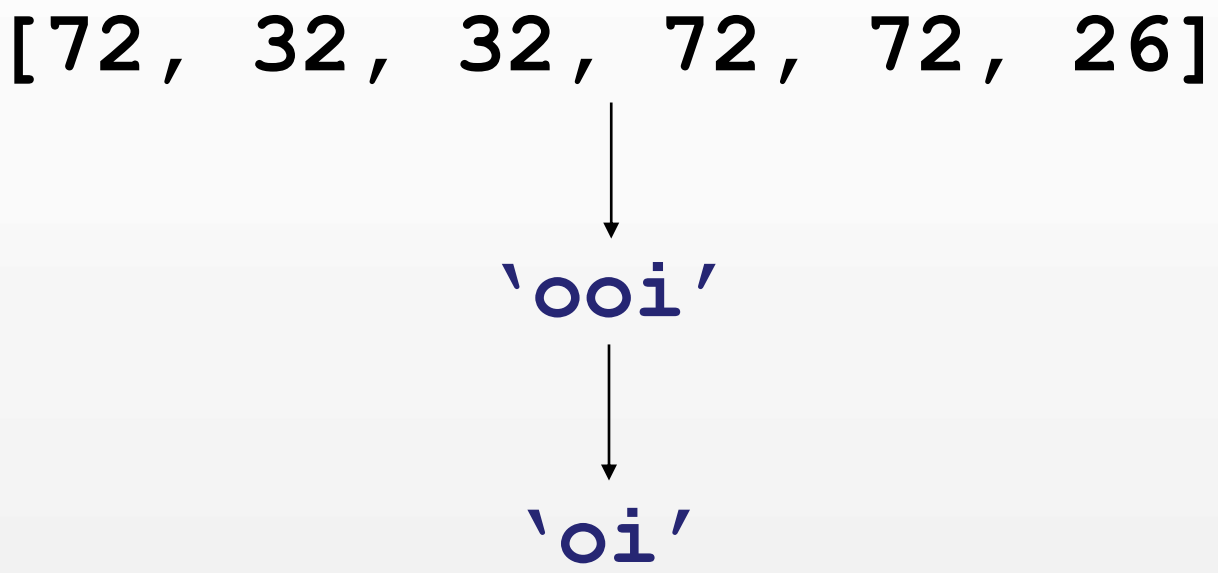
# Implementação

Treinamento

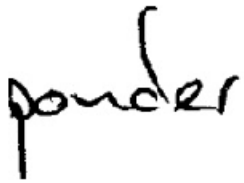
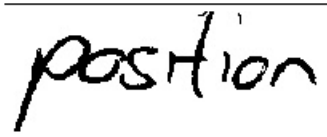
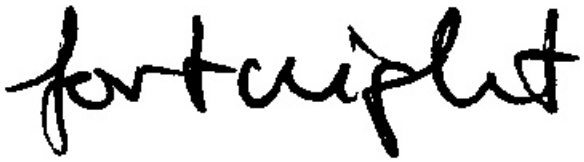
```
cost = self.ctc.log_ctc()
grads = tt.grad(cost, self.params)
updates = [
    (param_i, param_i - learning_rate * grad_i)
    for param_i, grad_i in zip(self.params, grads)
]

train_model = theano.function(
    inputs=[self.input, self.y],
    outputs=cost,
    updates=updates
)
```

# Implementação

Decodificação



# Resultados e Discussões

| Imagem | Rótulo | Previsão |
|---|---|---|
|  | ponder | poder |
|  | position | postion |
|  | atomic | atomic |
|  | to | to |
|  | reservations | heoservations |
|  | fortnight | tofortuight |

# Resultados e Discussões

- CER e WER
- Performance do protótipo:
  - WER: 71,4 %
  - CER: 48,38 %
- Palavras com menos de 4 caracteres:
  - WER: 58,62 %
  - CER: 48,83 %

# Resultados e Discussões

| Características / trabalhos correlatos | Graves (2012) | Su et al. (2015) | Chen, Yan e Huo (2015) | Caíque Reinhold (2016) |
|---|---|---|---|---|
| Técnicas de pré-processamento | Binarização | Correção das inclinações vertical e horizontal e binarização | Normallização da altura e binarização | Normalização da altura e binarização |
| Mecanismo de classificação | RNRMD | BLSTM | LSTM + HMM | RNC + LSTM |
| Características utilizadas pelo classificador | Pixels da imagem | Características geométricas e HOG | *Principal Component Analysis* | Pixels da imagem |
| Tipo de texto reconhecido | Códigos postais | Formulários | Sem restrição | Sem restrição |
| WER | 10,58 | 6,7 | 29,03 | 71,4 |
| CER | Não informado | Não informado | 15,16 | 48,83 |

# Conclusões e Sugestões

- Alternativa válida para reconhecimento de texto manuscrito
- Pouco tempo de treinamento
- Necessário uma decodificação mais robusta

# Extensões

- Uso de LSTM bidirecional
- Uso de um algoritmo de decodificação mais robusto
- Utilização de técnicas de *overfitting*
- Desenvolvimento de um sistema de reconhecimento de texto manuscrito baseado no modelo
- Criação de uma base de dados de texto manuscrito em português