Tutorial de Deep Learning

Luiz Gustavo Hafemann
LIVIA
École de Technologie Supérieure – Montréal

Organização do tutorial

- **Dia 1:**
  - Introdução à aprendizagem de máquina
  - Computação simbólica com Theano
- **Dia 2**
  - Redes neurais convolucionais

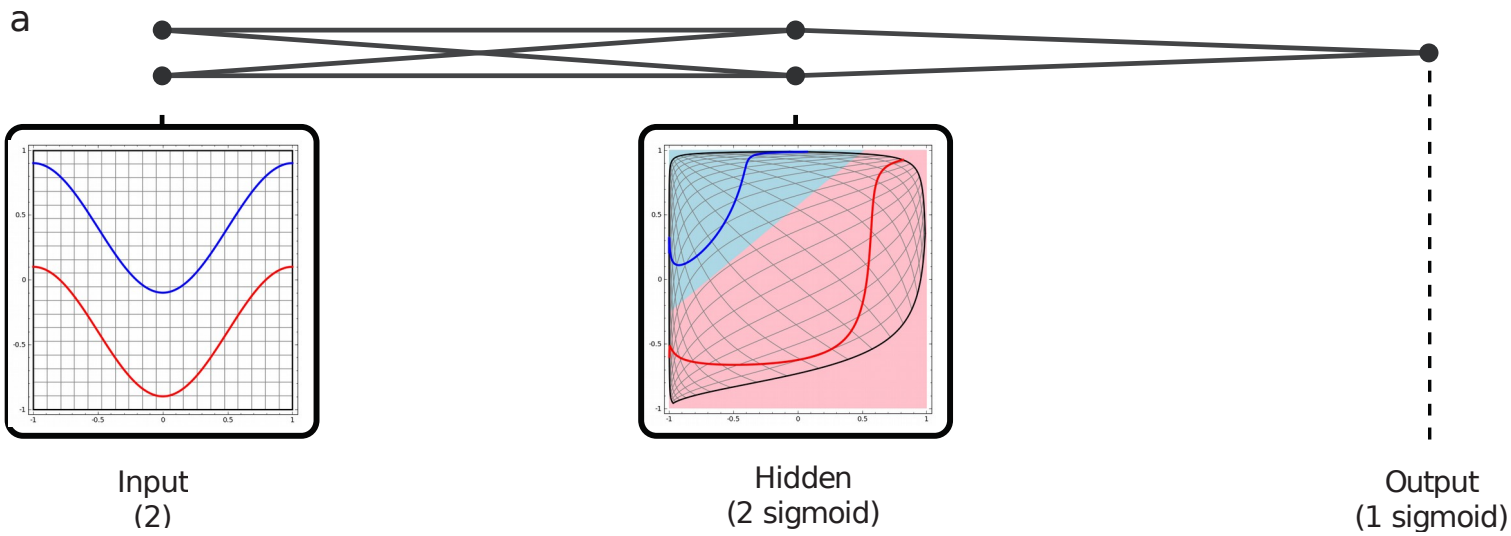
  Dia 3
  - Transfer Learning

Intuição de utilizar múltiplas camadas

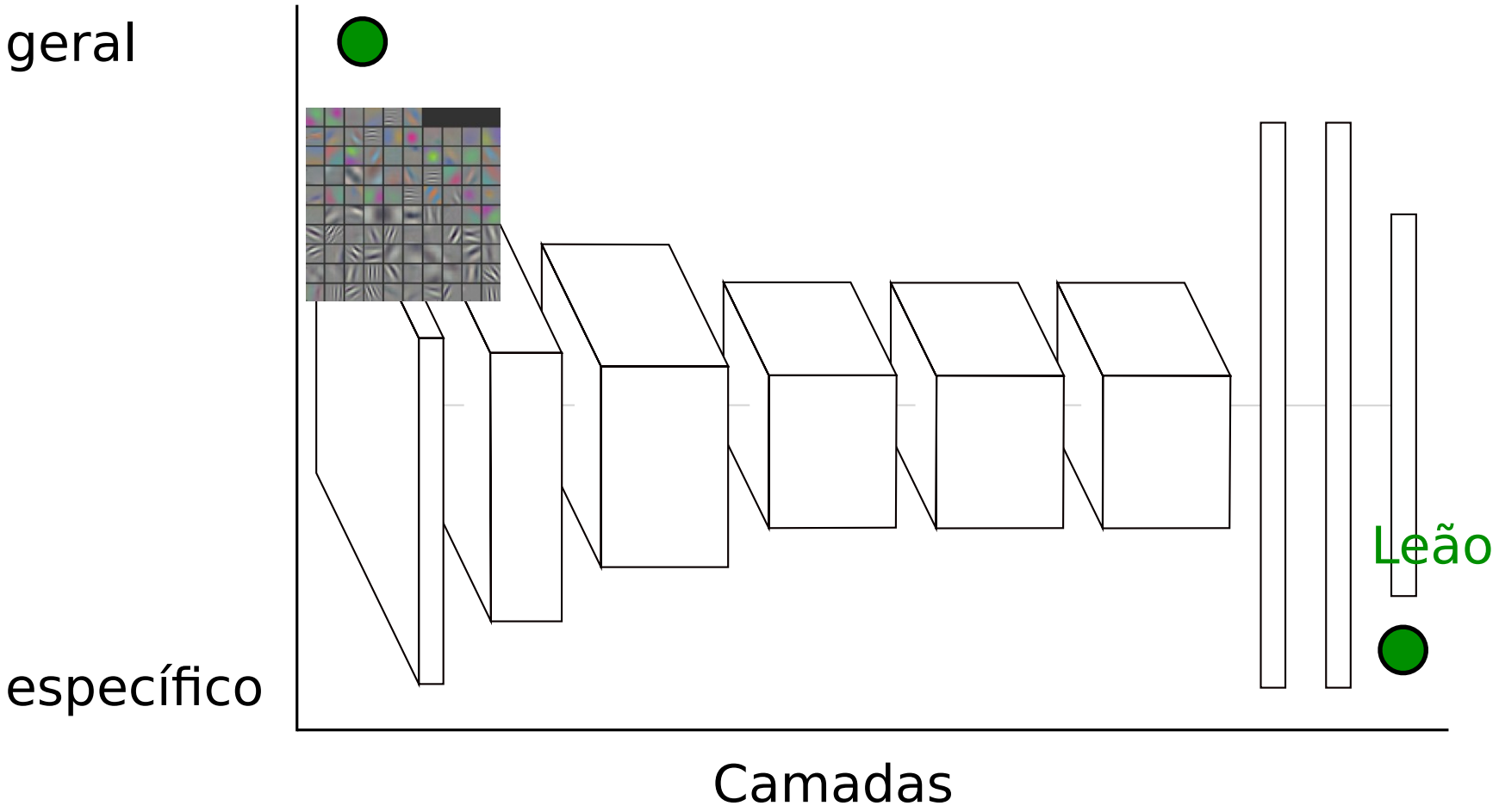
Intuição de utilizar múltiplas camadas

A última camada é um classificador linear dado a representação da penúltima camada (regressão logística)

Cada camada aprende uma transformação não-linear que "desentrelaça" as classes, de forma que sejam separadas linearmente



Representações aprendidas pela rede



(Yosinski, Jason, et al. "How transferable are features in deep neural networks?." 2014)

Representações aprendidas pela rede

Representações aprendidas pela rede

Primeiras camadas aprendem representações genéricas

Exemplo: detector de bordas na primeira camada

Representações aprendidas pela rede

Primeiras camadas aprendem representações genéricas

Exemplo: detector de bordas na primeira camada

Última camada aprende representação específica às classes

Representações aprendidas pela rede

Primeiras camadas aprendem representações genéricas

Exemplo: detector de bordas na primeira camada

Última camada aprende representação específica às classes

Representação em camadas intermediárias pode ser útil para outros problemas / bases de dados

Transfer Learning

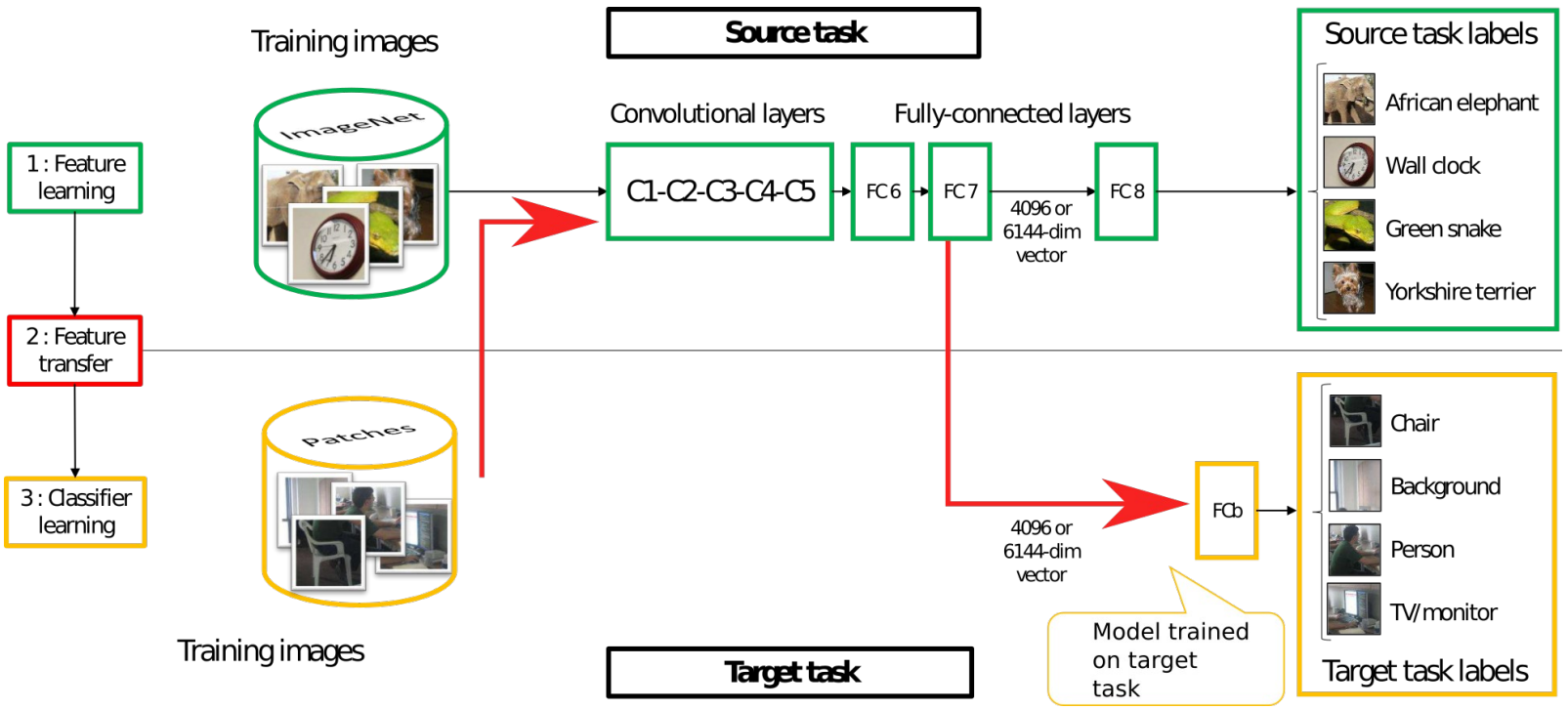
Conceito básico de Transfer Learning:

- Utilizar conhecimento aprendido em uma tarefa (origem) para melhor a performance em outra tarefa (destino)

Transferência de representações

- Utilizar uma representação aprendida em uma tarefa para melhorar a performance em outra tarefa

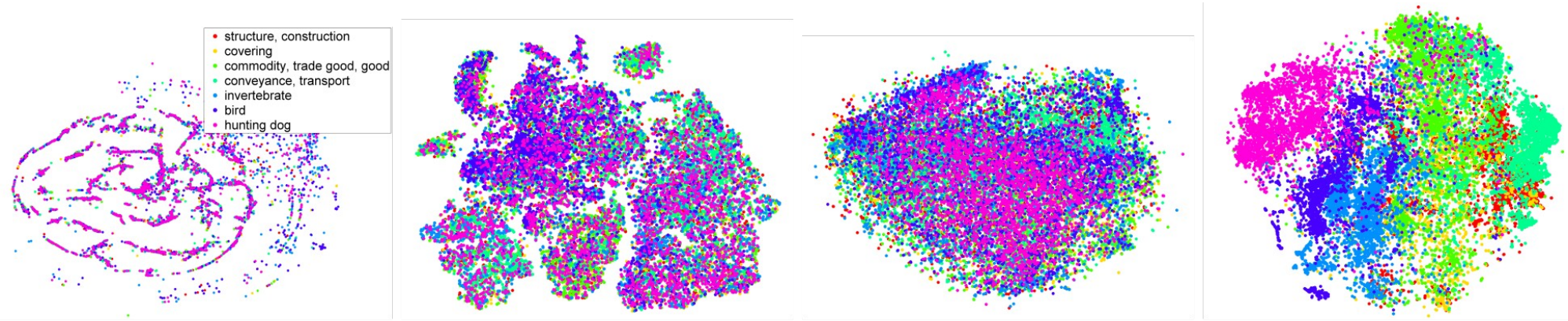
Transfer Learning - DeCAF



(Algoritmo: Jeff Donahue et al., "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition,", 2013; Imagem: Oquab et al.)

Visualizando o espaço de características

Visualizando como exemplos estão dispersos no espaço de características (usando t-SNE)



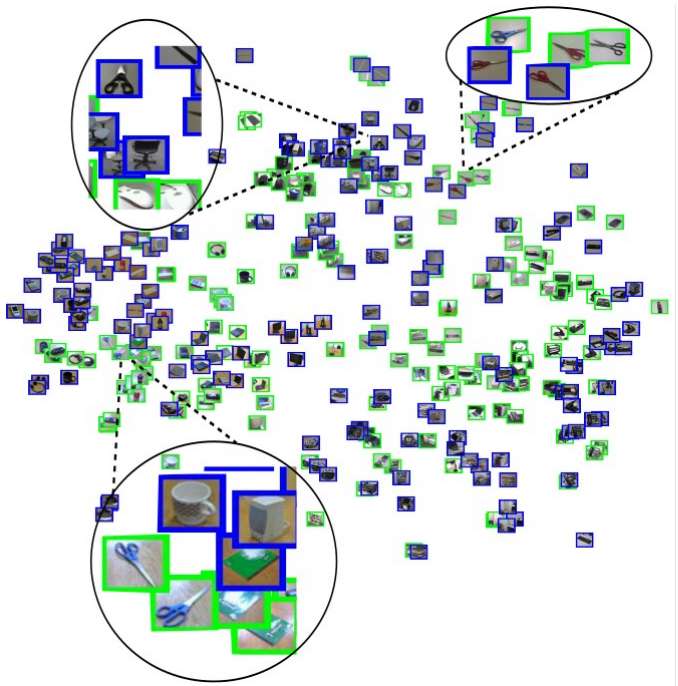
(a) LLC (b) GIST (c) DeCAF_1 (d) DeCAF_6

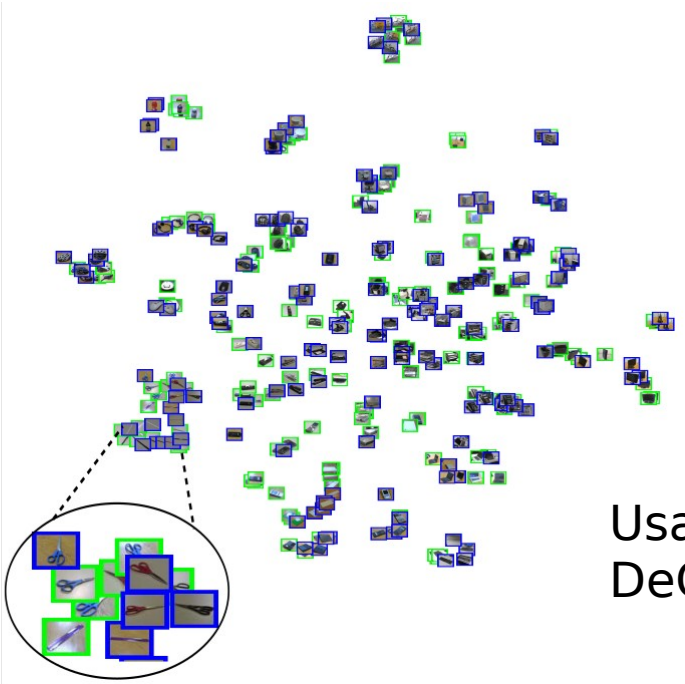
(Jeff Donahue et al., "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition,", 2013)

Visualizando o espaço de características

Usando a representação aprendida em uma base de dados, para dados de outra base:

Usando SURF





Usando DeCAF6

(Jeff Donahue et al., "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition,", 2013)

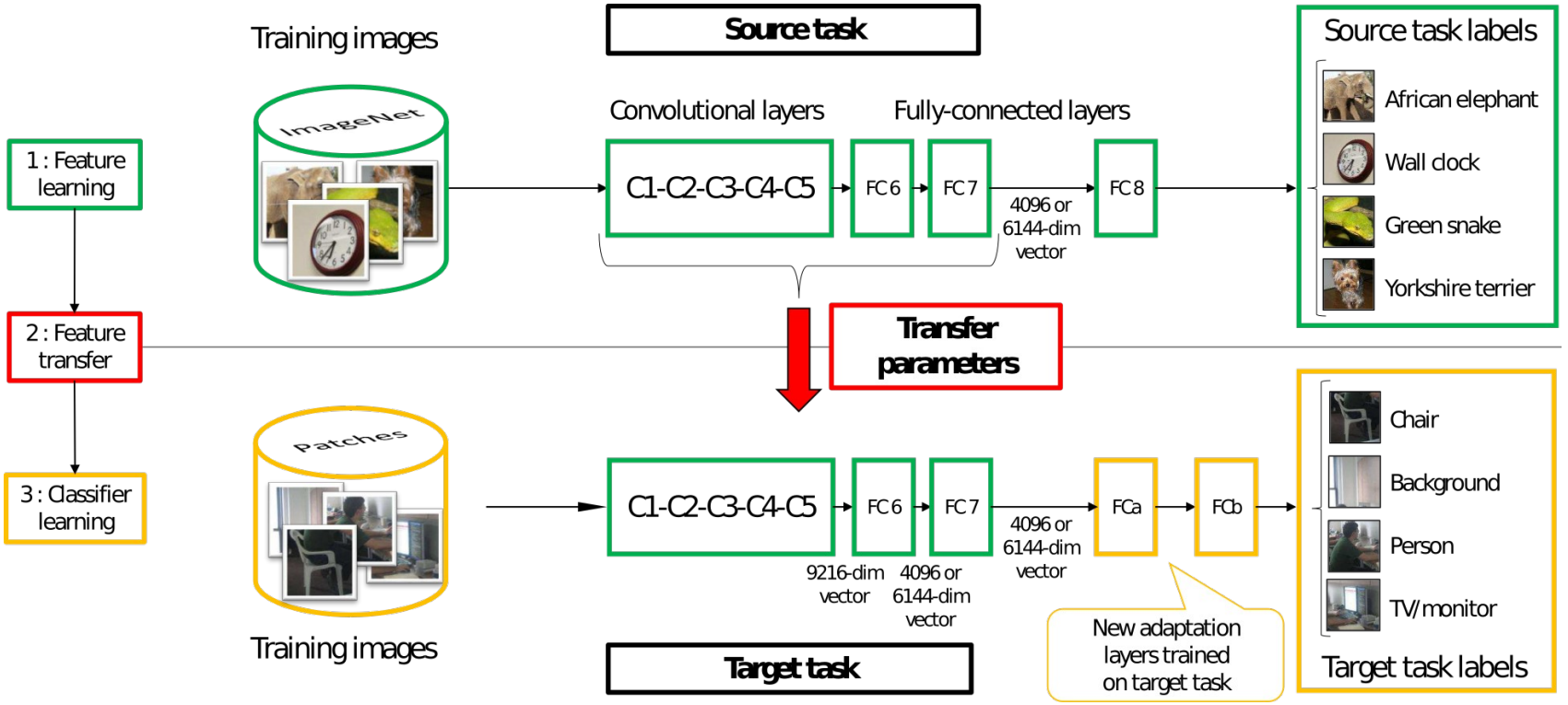
Transfer Learning

Método utilizado em Decaf:

- Treinar a rede em uma base de dados
- Utilizar a rede para "extrair características" de outra base de dados (fazendo forward-propagation e usando a ativação dos neurônios da camada)
- Treinar um classificador (SVM Linear) usando essa representação

=> Obteve estado-da-arte em vários datasets, utilizando rede aprendida na base ImageNet. Em particular, **em bases de dados pequenas**

Transfer learning



M. Oquab et al., "Learning and Transferring Mid-Level Image Representations Using Convolutional Neural Networks," in Computer Vision and Pattern Recognition

Transfer learning

Método utilizado por Oquab et al:

- Treinar a rede em uma base de dados (origem)
- Criar uma nova rede, com quase todas as camadas iguais, mas substituindo a última camada por uma ou mais camadas
- Treinar a rede na segunda base de dados (destino)
  - Opção 1: treinar toda a rede
  - Opção 2: treinar apenas as últimas camadas

Transfer Learning na prática

	Bases origem e destino parecidas	Bases origem e destino muito diferentes
Muitos poucos dados na base de destino		
Quantidade razoável de dados na base de destino		

Slide de Karpathy (CS231n)

Transfer Learning na prática

	Bases origem e destino parecidas	**Bases origem e destino muito diferentes**
Muitos poucos dados na base de destino	Treinar um classificador linear usando a representação da última camada	
Quantidade razoável de dados na base de destino	Re-treinar (finetune) algumas das últimas camadas	

Slide de Karpathy (CS231n)

Transfer Learning na prática

	Bases origem e destino parecidas	**Bases origem e destino muito diferentes**
Muitos poucos dados na base de destino	Treinar um classificador linear usando a representação da última camada	Não necessariamente funcione – tente treinar classificadores usando representações de diferentes camadas
Quantidade razoável de dados na base de destino	Re-treinar (finetune) algumas das últimas camadas	Re-treinar (finetune) várias (out todas as) camadas

Slide de Karpathy (CS231n)

Estudo de caso – transfer learning

Problema: Verificação de assinaturas manuscritas

- Verificar a identidade de uma pessoa usando sua assinatura

- Duas fases:
  - Registro: Usuário providencia algumas assinaturas genuínas
  - Operação: Uma pessoa providencia uma assinatura e diz ser determinado usuário. Objetivo do sistema: classificar a assinatura em genuína (produzida pelo usuário desejado) ou falsificação.

Verificação de assinaturas manuscritas

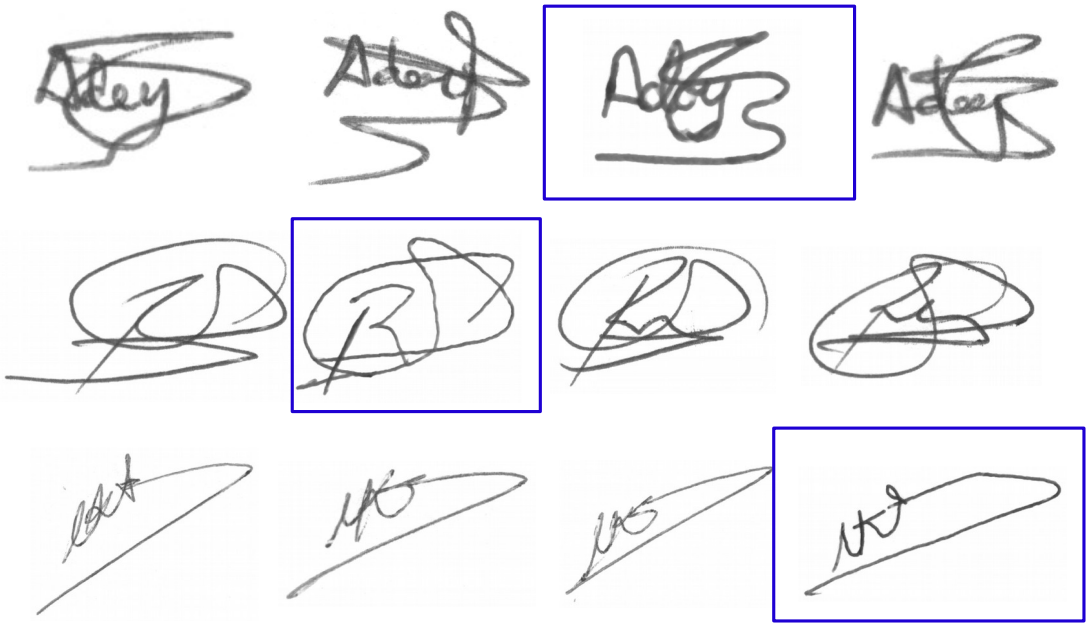
Desafios:

- Possuímos apenas assinaturas genuínas para treinamento
- Poucas assinaturas para treinamento (e.g. 3 – 10)
- Particularmente difícil discriminar falsificações exercitadas

Verificação de assinaturas manuscritas

Desafios:

- Possuímos apenas assinaturas genuínas para treinamento
- Poucas assinaturas para treinamento (e.g. 3 – 10)
- Particularmente difícil discriminar falsificações exercitadas



Verificação de assinaturas manuscritas

Como aprender representações?

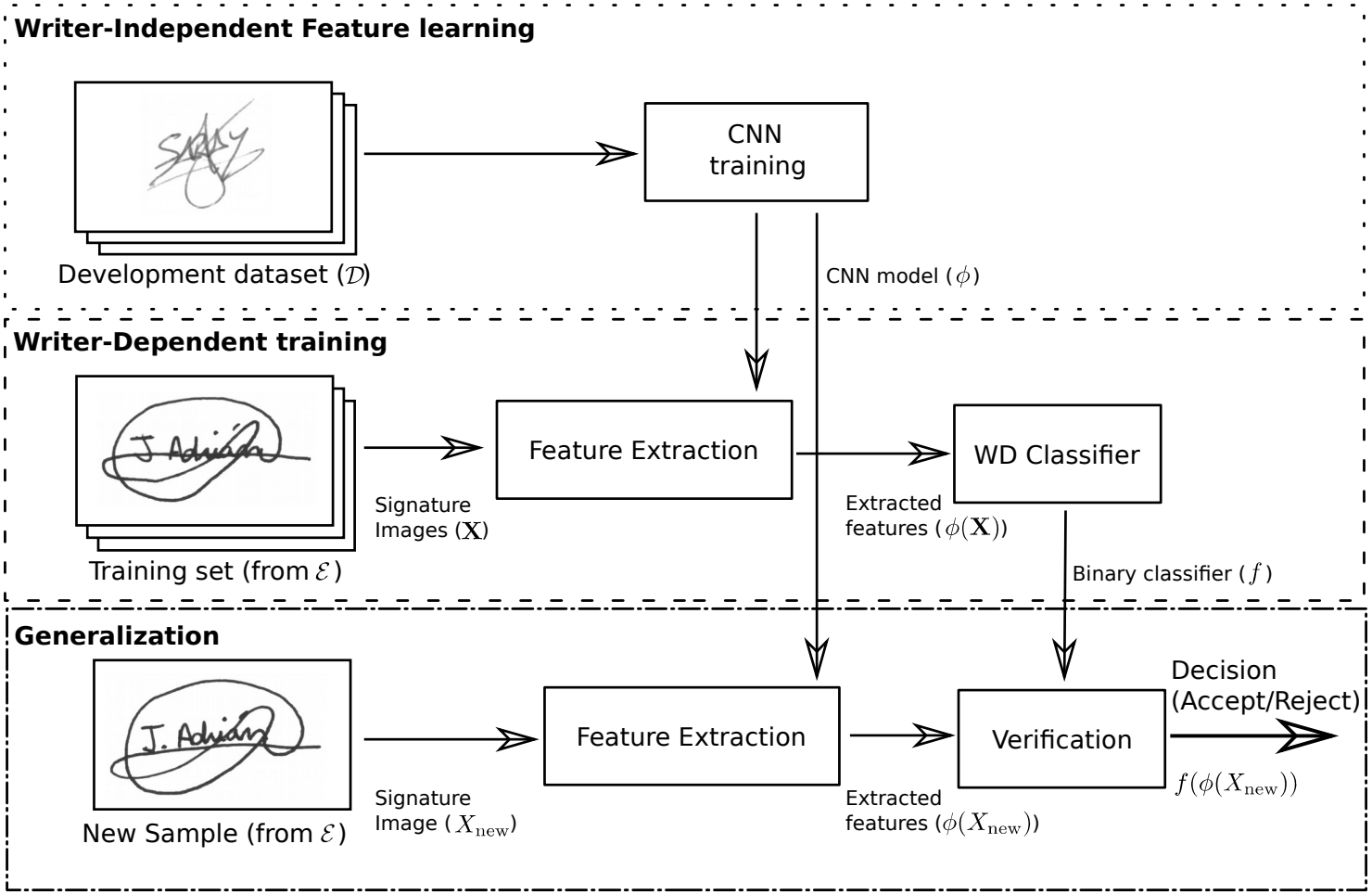
- Poucas assinaturas por usuários: praticamente impossível de aprender uma CNN para cada usuário
- Lista de usuários não é fixa
- Não podemos modelar o problema que realmente queremos resolver (separar assinaturas genuínas de falsificações), pois não temos falsificações para treinamento.

Verificação de assinaturas manuscritas

Solução proposta

- Aprender características que sejam independente do usuário (i.e. generalizem para qualquer usuário)
- Em seguida, treinar um classificador para cada usuário

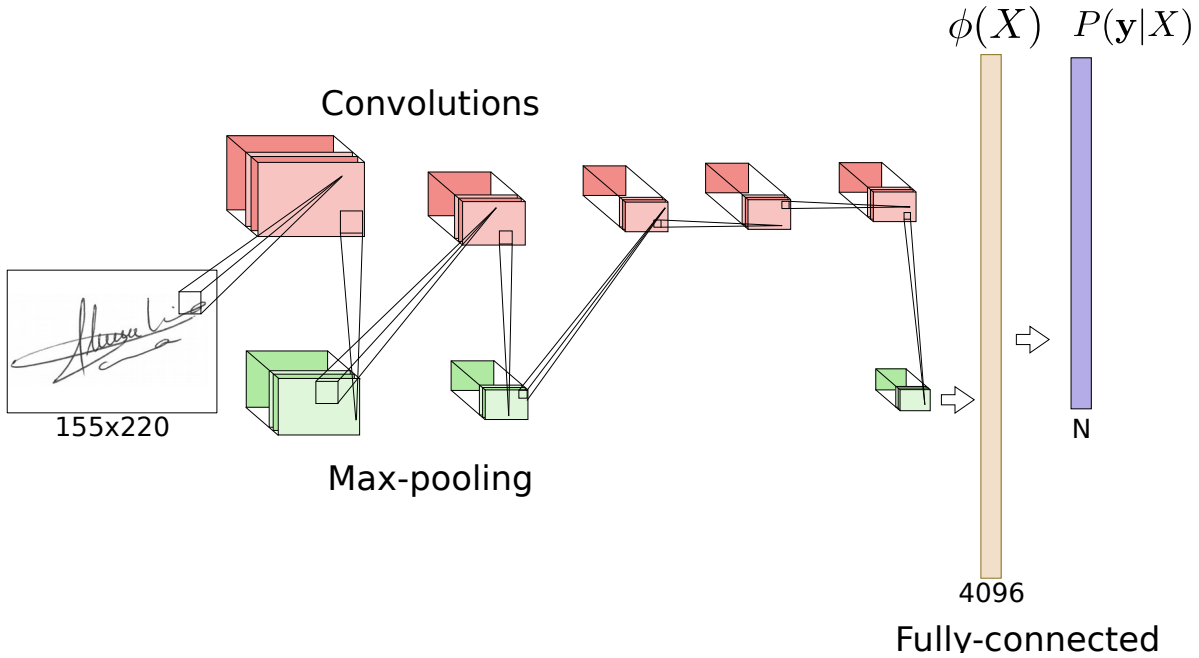
Solução proposta



Hafemann, Luiz G., Robert Sabourin, and Luiz S. Oliveira. "Writer-independent Feature Learning for Offline Signature Verification using Deep Convolutional Neural Networks."

Treinamento da CNN

- **Discriminar entre diferentes usuários:**
  - Entrada: imagens de assinaturas
  - Saída da CNN: $P(\mathbf{y}|X)$
  - Treinamento: Minimizar cross-entropy $-\sum_i \log(P(\mathbf{y}^{(i)}|X^{(i)}))$



Transfer learning

- **Para cada usuário:**
  - Criar uma base de treinamento com r assinaturas genuínas como pontos positivos, e assinaturas de outros usuários como pontos negativos
  - Para cada assinatura X, calcular $\phi(X)$: Forward propagation na CNN até a penúltima camada
  - Utilizar esses vetores de características para treinar um classificador binário: f
  - Para um novo exemplo X_{new} calcular: $f(\phi(X_{\text{new}}))$

Resultados

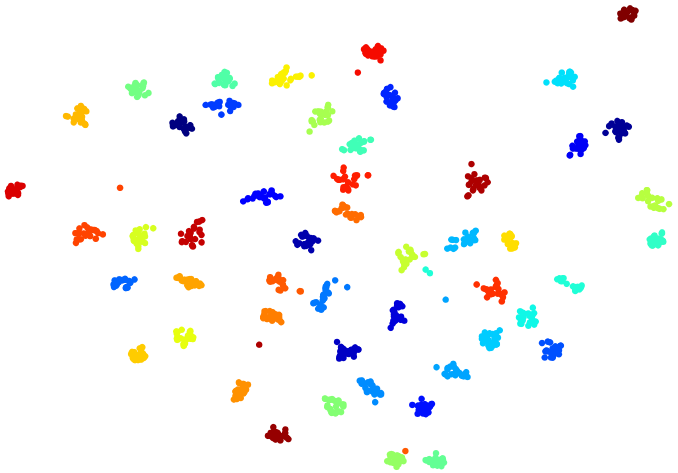
Comparação com estado da arte

base GPDS (831 usuários)

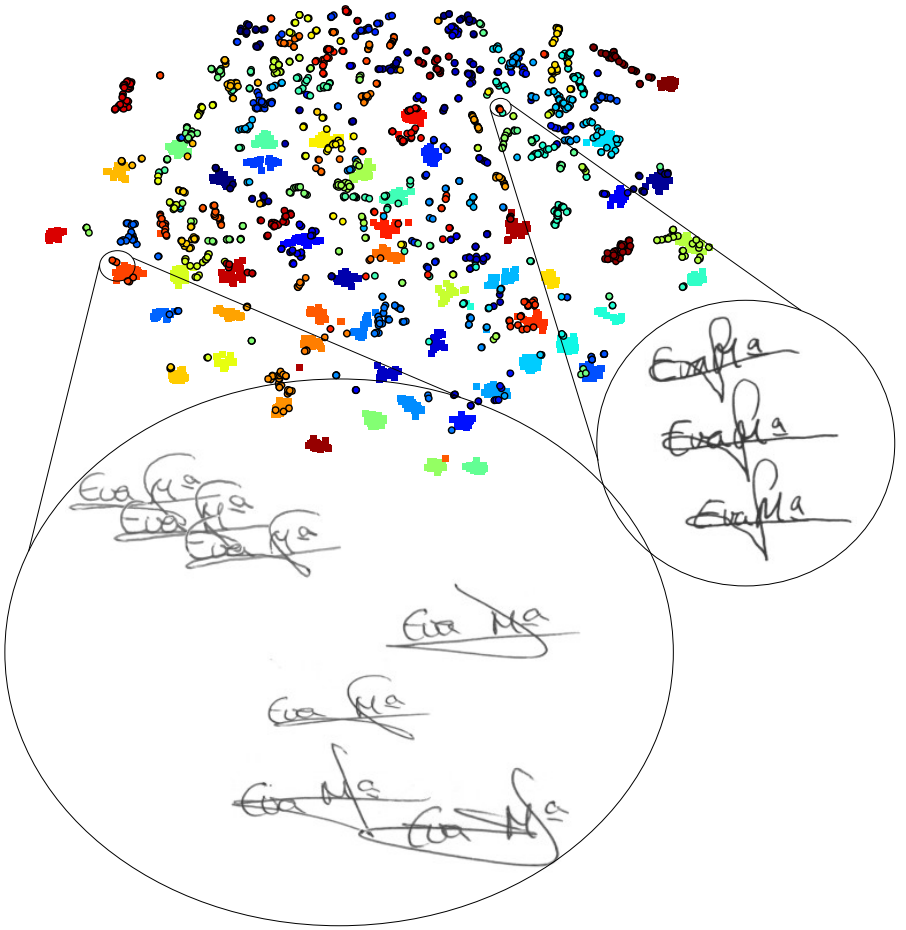
Reference	Dataset	#samples per user	Features & (Classif er)	EER
Vargas et al [19]	GPDS-100	5	Wavelets (SVM)	14.22
Vargas et al [20]	GPDS-100	10	LBP, GLCM (SVM)	9.02
Hu and Chen [7]	GPDS-150	10	LBP, GLCM, HOG (Adaboost)	7.66
Yilmaz [6]	GPDS-160	12	LBP (SVM)	9.64
Yilmaz [6]	GPDS-160	12	LBP, HOG (Ensemble of SVMs)	6.97
Hafemann et al [9]	GPDS-160	14	WI-learned with a CNN (SVM)	10.70
Present work	GPDS-160	5	WI-learned with a CNN (SVM)	3.83 (+- 0.33)
Present work	GPDS-160	14	WI-learned with a CNN (SVM)	2.74 (+- 0.18)
Present work	GPDS-300	5	WI-learned with a CNN (SVM)	4.53 (+- 0.14)
Present work	GPDS-300	14	WI-learned with a CNN (SVM)	3.47 (+- 0.16)

Hafemann, Luiz G., Robert Sabourin, and Luiz S. Oliveira. "Analyzing features learned for Offline Signature Verification using Deep CNNs."

Visualização do espaço de características



Assinaturas genuínas



Genuínas e falsificações

Transfer learning usando Lasagne

Método proposto em Decaf:

- Considere que temos uma rede já treinada em uma base de dados origem
- Precisamos de um método para obter a representação em uma camada, dado uma entrada X
- Trivial de ser implementado em Lasage:

Transfer learning usando Lasagne

Método proposto em Decaf:

- Considere que temos uma rede já treinada em uma base de dados origem
- Precisamos de um método para obter a representação em uma camada, dado uma entrada X
- Trivial de ser implementado em Lasage:

```
output_at_fc6 = lasagne.layers.get_output(net['fc6'], input_var,
```

Transfer learning usando Lasagne

Método proposto em Decaf:

- Considere que temos uma rede já treinada em uma base de dados origem
- Precisamos de um método para obter a representação em uma camada, dado uma entrada X
- Trivial de ser implementado em Lasage:

```
output_at_fc6 = lasagne.layers.get_output(net['fc6'], input_var,
                                          deterministic=True)
```

Transfer learning usando Lasagne

Método proposto em Decaf:

- Considere que temos uma rede já treinada em uma base de dados origem
- Precisamos de um método para obter a representação em uma camada, dado uma entrada X
- Trivial de ser implementado em Lasage:

```
output_at_fc6 = lasagne.layers.get_output(net['fc6'], input_var,
                                          deterministic=True)
get_fc6 = theano.function(input_var, output_at_fc6)
```

Transfer learning usando Lasagne

Método proposto em Decaf:

- Considere que temos uma rede já treinada em uma base de dados origem
- Precisamos de um método para obter a representação em uma camada, dado uma entrada X
- Trivial de ser implementado em Lasage:

```
output_at_fc6 = lasagne.layers.get_output(net['fc6'], input_var,
                                          deterministic=True)
get_fc6 = theano.function(input_var, output_at_fc6)

new_X = get_fc6(X)
```

Transfer learning usando Lasagne

Método de fine-tuning (re-treinamento):

- Considere que temos uma rede já treinada em uma base de dados origem
- Precisamos:
  - Criar uma rede com a mesma arquitetura, com exceção da última camada
  - Copiar os pesos da rede antiga para a rede nova
  - Efetuar o treinamento com a base de dados destino

Transfer learning usando Lasagne

Transfer learning usando Lasagne

```
model = build_model() #Constrói o modelo da base origem
```

Transfer learning usando Lasagne

```
model = build_model() #Constrói o modelo da base origem
lasagne.layers.set_all_param_values(model['out'], params)  #Copia parametros
```

Transfer learning usando Lasagne

```
model = build_model() #Constrói o modelo da base origem
lasagne.layers.set_all_param_values(model['out'], params)  #Copia parametros

del model['out']     # Deleta a última camada
```

Transfer learning usando Lasagne

```
model = build_model() #Constrói o modelo da base origem
lasagne.layers.set_all_param_values(model['out'], params)  #Copia parametros

del model['out']     # Deleta a última camada

#criando uma (ou mais) camadas:
```

Transfer learning usando Lasagne

```
model = build_model() #Constrói o modelo da base origem
lasagne.layers.set_all_param_values(model['out'], params)  #Copia parametros

del model['out']     # Deleta a última camada

#criando uma (ou mais) camadas:
model['out'] = DenseLayer(model['fc7'], nClasses, nonlinearity=softmax)
```

Transfer learning usando Lasagne

Transfer learning usando Lasagne

Treinando apenas algumas camadas:

Se quisermos limitar o treinamento à apenas algumas camadas, precisamos alterar a seguinte parte da função de treinamento:

Transfer learning usando Lasagne

Treinando apenas algumas camadas:

Se quisermos limitar o treinamento à apenas algumas camadas, precisamos alterar a seguinte parte da função de treinamento:

```
params = lasagne.layers.get_all_params(net['out'])
```

Transfer learning usando Lasagne

Treinando apenas algumas camadas:

Se quisermos limitar o treinamento à apenas algumas camadas, precisamos alterar a seguinte parte da função de treinamento:

```
params = lasagne.layers.get_all_params(net['out'])
updates = lasagne.updates.sgd(loss, params, lr)
```

Transfer learning usando Lasagne

Treinando apenas algumas camadas:

Se quisermos limitar o treinamento à apenas algumas camadas, precisamos alterar a seguinte parte da função de treinamento:

```
params = lasagne.layers.get_all_params(net['out'])
updates = lasagne.updates.sgd(loss, params, lr)
```

Para cada camada que queremos treinar, obtemos os parâmetros usando “camada.get_params”:

Transfer learning usando Lasagne

Treinando apenas algumas camadas:

Se quisermos limitar o treinamento à apenas algumas camadas, precisamos alterar a seguinte parte da função de treinamento:

```
params = lasagne.layers.get_all_params(net['out'])
updates = lasagne.updates.sgd(loss, params, lr)
```

Para cada camada que queremos treinar, obtemos os parâmetros usando "camada.get_params":

```
params = []
```

Transfer learning usando Lasagne

Treinando apenas algumas camadas:

Se quisermos limitar o treinamento à apenas algumas camadas, precisamos alterar a seguinte parte da função de treinamento:

```
params = lasagne.layers.get_all_params(net['out'])
updates = lasagne.updates.sgd(loss, params, lr)
```

Para cada camada que queremos treinar, obtemos os parâmetros usando "camada.get_params":

```
params = []
for l in layers_to_train:
```

Transfer learning usando Lasagne

Treinando apenas algumas camadas:

Se quisermos limitar o treinamento à apenas algumas camadas, precisamos alterar a seguinte parte da função de treinamento:

```
params = lasagne.layers.get_all_params(net['out'])
updates = lasagne.updates.sgd(loss, params, lr)
```

Para cada camada que queremos treinar, obtemos os parâmetros usando "camada.get_params":

```
params = []
for l in layers_to_train:
    params += l.get_params(trainable=True)
```

Transfer learning usando Lasagne

Treinando apenas algumas camadas:

Se quisermos limitar o treinamento à apenas algumas camadas, precisamos alterar a seguinte parte da função de treinamento:

```
params = lasagne.layers.get_all_params(net['out'])
updates = lasagne.updates.sgd(loss, params, lr)
```

Para cada camada que queremos treinar, obtemos os parâmetros usando "camada.get_params":

```
params = []
for l in layers_to_train:
    params += l.get_params(trainable=True)
updates = lasagne.updates.sgd(loss, params, lr)
```

Exercícios

Transfer Learning - Parte 1.ipynb

Transfer Learning - Parte 2.ipynb