MBA em Ciência de Dados

Redes Neurais e Arquiteturas Profundas

Módulo IV - Estratégias de Treinamento e Transferência de Aprendizado

Avaliação (com soluções)

Moacir Antonelli Ponti

CeMEAI - ICMC/USP São Carlos

As respostas devem ser dadas no Moodle, use esse notebook apenas para gerar o código necessário para obter as respostas

---

Questão 1)

Qual a relação entre o modelo chamado de "memorizador" e as redes neurais profundas?

(a) Redes neurais com alta capacidade podem memorizar todos os exemplos de treinamento, tornando-as hábeis para generalizar para dados futuros.
(b) Redes neurais com alta capacidade podem memorizar todos os exemplos de treinamento, falhando em predizer corretamente exemplos não vistos.
(c) Redes neurais com alta capacidade são imunes a convergir para modelos memorizadores, pois obtiveram resultados do estado-da-arte em muitas aplicações.
(d) Redes neurais com alta capacidade podem memorizar todos os exemplos de treinamento, e portanto possuem viés forte.

Justificativa: O modelo "memória´´ ou memorizador é considerado o caso extremo de overfitting em que o modelo tem custo zero no treinamento, mas falha em obter predição em qualquer exemplo que fujam aos vistos no treinamento. Redes neurais profundas possuem tipicamente grande quantidade de parâmetros, podendo convergir para modelos desse tipo.

Questão 2)

O papel do uso conjunto dos métodos BatchNormalization e Regularização é o de:

(a) Pré-processamento dos dados antes da realização do treinamento
(b) Gerar espaço de parâmetros esparsos, com alguns poucos parâmetros com valor alto e muitos com valores próximo a zero, melhorando a generalização
(c) Minimizar o problema do desaparecimento do gradiente, e ao mesmo tempo evitar que poucas unidades/neurônios se especializem demais
(d) Obter robustez com relação à possíveis ataques e propiciar modelos menores com

acurácia similar a modelos maiores

Justificativa: Métodos de normalização auxiliam no treinamento suavizando o gradiente (e evitando seu desaparecimento), e de regularização evitam especialização de poucos pesos/neurônios.

Questão 3)

São práticas viáveis para o uso de aprendizado profundo com pequenas bases de dados:

(a) Carregar uma rede neural profunda popular de um pacote de software e treiná-la a partir de pesos aleatórios
(b) Carregar uma rede neural profunda pré-treinada em grande base de dados, e utilizar a saída da última camada da rede, ou seja as predições das classes, como característica para modelos de aprendizado externos que permitem uso com menores bases de dados
(c) Carregar uma rede neural profunda popular de um pacote de software e treiná-la a partir de pesos aleatórios utilizando Batch Normalization
(d) Carregar uma rede neural profunda pré-treinada em grande base de dados, inserindo uma nova camada de saída treinando apenas essa camada com a pequena base de dados

Justificativa: dentre as opções, treinar a partir de pesos aleatórios é comumente inviável em bases pequenas, além disso, apesar de possível, o uso da ativação da última camada da rede (predição das classes) não é recomendado pois traz semântica específica do conjunto de dados com o qual foi treinado.

Questão 4)

Carregue a base de dados Fashion MNIST conforme código abaixo e crie um modelo de CNN com a seguinte arquitetura, capaz de obter classificação nessa base de dados de imagens. Considere que todas as camadas convolucionais tem zeropadding, e ativação relu, exceto quando mencionado contrário.

1. Pré-processamento para aumentação contendo:
    - RandomZoom(0.1),
    - RandomContrast(0.2)
2. Convolucional 2D com 64 filtros 3×3.
3. Batch Normalization
4. SeparableConv2D com 64 filtros 3×3.
5. MaxPooling2D 3×3 e strides 2
6. Batch Normalization
7. SeparableConv2D com 256 filtros 3×3.
8. MaxPooling2D 3×3 e strides 2
9. GlobalAveragePooling
10. Dropout de 0.2
11. Densa com ativação softmax

Incialize as sementes do numpy com 1 e tensorflow com 2 e treine o modelo por 7 épocas com batch size 16, otimizador Adam e taxa de aprendizado 0.002.

Após o treinamento utilize a função predict para classificar imagens da posicao 10 a 14 no conjunto de testes ([10:15]). Quais as classes resultantes e quantas dessas estavam erradas?

(a) 2, 5, 5, 3, 3, sendo 2 erradas
(b) 4, 5, 5, 3, 4, sendo 2 erradas
(c) 4, 5, 5, 3, 4 sendo 1 errada
(d) 4, 5, 5, 3, 4, nenhuma errada

Justificativa: Ver código abaixo

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from numpy.random import seed
from tensorflow.random import set_seed

fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.lo

train_images = train_images / 255.0
test_images = test_images / 255.0

train_labels = keras.utils.to_categorical(train_labels, 10)
test_labels = keras.utils.to_categorical(test_labels, 10)
```

In [6]:
```
data_augmentation = keras.Sequential(
    [
        layers.experimental.preprocessing.RandomZoom(0.1),
        layers.experimental.preprocessing.RandomContrast(0.2),
    ]
)

def my_cnn(input_shape, num_classes, dropout_rate=0.0):

    inputs = keras.Input(shape=input_shape)
    x = data_augmentation(inputs)
    x = layers.Conv2D(64, 3, activation="relu", padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.SeparableConv2D(64, 3, activation="relu", padding="same")(x
    x = layers.MaxPooling2D(3, strides=2, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.SeparableConv2D(256, 3, activation="relu", padding="same")(x
    x = layers.MaxPooling2D(3, strides=2, padding="same")(x)
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dropout(dropout_rate)(x)
    outputs = layers.Dense(num_classes, activation="softmax")(x)

    return keras.Model(inputs, outputs)
```

In [7]:
```
seed(1)
set_seed(2)
epochs = 7
batch_size=16
model1 = my_cnn((28,28,1), 10, 0.2)
model1.compile(loss='categorical_crossentropy',
               optimizer=keras.optimizers.Adam(lr=0.002),
               metrics=['accuracy'])

hist1 = model1.fit(train_images, train_labels, batch_size=batch_size,
                   epochs=epochs, verbose=1)
```

```
Epoch 1/7
3750/3750 [==============================] - 105s 28ms/step - loss: 0.5496
- accuracy: 0.7965
Epoch 2/7
3750/3750 [==============================] - 102s 27ms/step - loss: 0.3869
- accuracy: 0.8577
Epoch 3/7
3750/3750 [==============================] - 97s 26ms/step - loss: 0.3473 -
accuracy: 0.8738
Epoch 4/7
3750/3750 [==============================] - 105s 28ms/step - loss: 0.3273
- accuracy: 0.8801
Epoch 5/7
3750/3750 [==============================] - 112s 30ms/step - loss: 0.3149
- accuracy: 0.8845
Epoch 6/7
3750/3750 [==============================] - 109s 29ms/step - loss: 0.3037
- accuracy: 0.8881
Epoch 7/7
3750/3750 [==============================] - 105s 28ms/step - loss: 0.2950
- accuracy: 0.8928
```

In [8]:
```
scores = model1.evaluate(train_images, train_labels)
print(scores)
```

```
1875/1875 [==============================] - 22s 12ms/step - loss: 0.2414 -
accuracy: 0.9110
[0.24139392375946045, 0.9110333323478699]
```

In [9]:
```
y_pred = model1.predict(test_images[10:15])
print(np.argmax(y_pred,1))
print(np.argmax(test_labels[10:15],1))

#[4 5 5 3 4]
#[4 5 7 3 4]
#print(np.round(np.max(y_pred,1),1))
```

```
[4 5 5 3 4]
[4 5 7 3 4]
```

In [10]:
```
model1.summary()
```

```
Model: "functional_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
sequential_2 (Sequential)    (None, 28, 28, 1)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 64)        640
_________________________________________________________________
```

```
batch_normalization_3 (Batch (None, 14, 14, 64)        256
_________________________________________________________________
separable_conv2d_4 (Separabl (None, 14, 14, 64)        4736
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
batch_normalization_4 (Batch (None, 7, 7, 64)          256
_________________________________________________________________
separable_conv2d_5 (Separabl (None, 7, 7, 256)         17216
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 4, 4, 256)         0
_________________________________________________________________
global_average_pooling2d_2 ( (None, 256)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 25,674
Trainable params: 25,418
Non-trainable params: 256
```

Questão 5)

Carregue a base de dados MNIST do pacote Keras, e pre-processe conforme código abaixo.

Vamos utilizar o modelo treinado na questão anterior como forma de trasnferência de aprendizado. Se preciso reinicialize o modelo e treine-o novamente para garantir que apenas 7 épocas foram executadas. O modelo final deve ter acurácia de treinamento próxima a 0.89 (computada na base Fashion).

Agora, assuma que esse modelo já treinado está armazenado numa variável `model` . Então proceda da seguinte forma:

1. Obtendo a saída da penúltima camada (referente ao Dropout): `base_saida = model.layers[-2].output`
2. Criando uma nova camada de saída que recebe como entrada a anterior `saida_nova = keras.layers.Dense(10, activation='softmax')(base_saida)`
3. Criando um novo modelo tendo essa nova camada como saída `model2 = keras.models.Model(model.inputs, saida_nova)`

Você pode usar o summary para conferir o modelo montado.

Agora inicialize as sementes do numpy para 1 e tensorflow para 2, compile e treine o novo modelo com função de custo entropia cruzada categórica, otimizador Adam com taxa de aprendizado 0.002, 16 exemplos no mini-batch e 3 épocas.

Avalie a acurácia no conjunto de testes. Em qual intervalo está a acurácia resultante, considerando arredondamento para 2 casas decimais?

(a) [0.94,0.96]
(b) [0.98,1.00]

(c) [0.87,0.90]

(d) [0.92,0.93]

Justificativa: Ver código abaixo.

In [11]:
```
mnist = keras.datasets.mnist
(train_images2, train_labels2), (test_images2, test_labels2) = mnist.load_
train_images2 = train_images2 / 255.0
test_images2 = test_images2 / 255.0
train_labels2 = keras.utils.to_categorical(train_labels2, 10)
test_labels2 = keras.utils.to_categorical(test_labels2, 10)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-da
tasets/mnist.npz
11493376/11490434 [==============================] - 1s 0us/step
```

In [12]:
```
base_output = model1.layers[-2].output
newout = keras.layers.Dense(10, activation='softmax')(base_output)
model2 = keras.models.Model(model1.inputs, newout)
model2.summary()

seed(1)
set_seed(2)
model2.compile(loss='categorical_crossentropy',
               optimizer=keras.optimizers.Adam(lr=0.002),
               metrics=['accuracy'])

hist2 = model2.fit(train_images2, train_labels2, batch_size=16,
                   epochs=3, verbose=1)
```

```
Model: "functional_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
sequential_2 (Sequential)    (None, 28, 28, 1)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 64)        640
_________________________________________________________________
batch_normalization_3 (Batch (None, 14, 14, 64)        256
_________________________________________________________________
separable_conv2d_4 (Separabl (None, 14, 14, 64)        4736
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
batch_normalization_4 (Batch (None, 7, 7, 64)          256
_________________________________________________________________
separable_conv2d_5 (Separabl (None, 7, 7, 256)         17216
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 4, 4, 256)         0
_________________________________________________________________
global_average_pooling2d_2 ( (None, 256)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                2570
=================================================================
Total params: 25,674
Trainable params: 25,418
Non-trainable params: 256
_________________________________________________________________
Epoch 1/3
```

```
3750/3750 [==============================] - 107s 29ms/step - loss: 0.1635
- accuracy: 0.9496
Epoch 2/3
3750/3750 [==============================] - 113s 30ms/step - loss: 0.0681
- accuracy: 0.9783
Epoch 3/3
3750/3750 [==============================] - 115s 31ms/step - loss: 0.0572
```

In [13]:
```
scores = model2.evaluate(test_images2, test_labels2, verbose=0)
print("Acurácia teste: %.2f" % (scores[1]))
```

```
Acurácia teste: 0.98
```

In [7]:
```
```