Lista de Exercícios - Preparação para Prova
Visão Computacional

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

Questão 1

Redes Neurais Convolucionais (ConvNets) são a arquitetura padrão para processamento de imagens. Modelos pioneiros como LeNet estabeleceram um *blueprint* arquitetônico que foi amplamente expandido por redes subsequentes, como AlexNet e VGG.

Assinale a alternativa que descreve corretamente a estrutura fundamental de um modelo ConvNet do tipo LeNet para classificação de imagens:

- A) Uma sucessão de camadas totalmente conectadas (Fully Connected Layers - FC) seguidas por uma camada de *pooling* e, finalmente, uma camada *softmax* para classificação.
- B) Uma série de camadas convolucionais 1x1 para redução da dimensionalidade, seguida por camadas recorrentes para captura de sequências.
- C) Uma série alternada de camadas convolucionais 2D e camadas de *max pooling*, atuando como extrator de características, seguida por camadas totalmente conectadas que realizam a classificação.
- D) Blocos de autoatenção (Self-Attention) que processam patches da imagem, sem a necessidade de camadas convolucionais tradicionais.
- E) Uso exclusivo de blocos residuais (*residual blocks*) para evitar o problema de gradiente evanescente, sem a inclusão de camadas de *pooling*.

Questão 2

As camadas convolucionais são o coração das ConvNets, oferecendo um *inductive bias* forte e benéfico para o processamento de imagens. Uma das principais vantagens dessas camadas é a sua relação com as transformações de translação (*translation*) na imagem.

Considerando um ConvNet para Visão Computacional:

I. Uma camada convolucional é **invariante à translação** se a translação da entrada resulta na translação da saída na mesma proporção. II. Uma camada convolucional é **equivariante à translação** se a translação da entrada resulta na translação da saída na mesma proporção. III. Para a tarefa de **classificação de imagem**, o objetivo é que a predição da rede seja **invariante à translação**, ou seja, a categoria deve ser a mesma, independentemente da posição do objeto na imagem.

É correto o que se afirma em:

- A) I, apenas.
- B) II, apenas.
- C) III, apenas.
- D) II e III, apenas.
- E) I, II e III.

Questão 3

Discuta a necessidade fundamental de incorporar **funções de ativação não-lineares** (como a ReLU ou Sigmoid) nas camadas ocultas de um Multi-Layer Perceptron (MLP). Explique como esta inclusão resolve a principal limitação inerente ao Perceptron Clássico de camada única.

Questão 4

Para a construção de um modelo de classificação em PyTorch, a escolha da **Função de Custo (Loss Function)** e da **Função de Ativação de Saída** é diretamente determinada pelo tipo de problema.

Detalhe as escolhas de **Função de Custo** e **Função de Ativação de Saída** para: 1. **Classificação Binária.** 2. **Classificação Multi-Classe.**

Em seguida, explique o processo de conversão das **saídas brutas do modelo (Logits)** em **rótulos de previsão finais** para *ambos* os casos, citando as funções PyTorch envolvidas.

Questão 5

O treinamento de uma Rede Neural é realizado por meio de um ciclo iterativo que depende fundamentalmente dos conceitos de *Forward Pass*, *Loss*, *Backpropagation* e *Otimização*.

Descreva os **passos essenciais** de um *PyTorch training loop*. Em seguida, explique o papel do **Gradiente Descendente (Gradient Descent)** na atualização dos parâmetros da rede, incluindo a função da **taxa de aprendizado** (η).

Questão 6:

Em um cenário de Visão Computacional, você é encarregado(a) de projetar uma Rede Neural Multi-Layer Perceptron (MLP) em PyTorch para a classificação de imagens. O dataset de entrada é composto por imagens em escala de cinza de **32x32 pixels**. A arquitetura da rede deve seguir as seguintes especificações rigorosas:

1. **Camada de Entrada:** Deve receber o vetor de pixels achatado (flattened) de cada imagem.
2. **Camadas Ocultas:** Duas (2) camadas ocultas, utilizando a função de ativação **ReLU** entre elas.
   - A primeira camada oculta deve possuir **20 neurônios**.
   - A segunda camada oculta deve possuir **10 neurônios**.
3. **Camada de Saída:** Deve ser dimensionada para um problema de **classificação multi-classe com 3 classes mutuamente exclusivas** (similar ao dataset MNIST).

Complete o **esboço de código Python/PyTorch** completo que defina a classe desta arquitetura de rede neural, incluindo a correta determinação da dimensão de entrada/saída e o método `forward()` com as funções de ativação especificadas.

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class ImageClassifierMLP(nn.Module):
    def __init__(self):
        super(ImageClassifierMLP, self).__init__()

        self.fc1 = nn.Linear(in_features=____________, out_features=____________)

        self.fc2 = nn.Linear(in_features=____________, out_features=____________)

        self.fc3 = nn.Linear(in_features=____________, out_features=____________)

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        # 1. Adicione aqui a instrução para transformar a imagem em um vetor de entrada para a MLP
```

```
        x = ____________________

        # 2. Camada Oculta 1 com Ativação ReLU
        x = ____________________

        # 3. Camada Oculta 2 com Ativação ReLU
        x = ____________________

        # 4. Camada de Saída (Output Logits)
        logits = ____________________

        return logits
```

Questão 7

O ajuste de um modelo paramétrico $f(\mathbf{x}, \phi)$ a um conjunto de dados de treinamento $\mathcal{D}$ envolve a minimização de uma função de perda $\mathcal{L}(\phi)$. Durante este processo, o desempenho do modelo deve ser monitorado tanto no conjunto de treinamento quanto em um conjunto de validação ou teste, pois o desempenho no treinamento pode ser enganoso.

Assinale a alternativa que descreve corretamente o diagnóstico e a causa fundamental dos fenômenos de *Underfitting* e *Overfitting*:

A) **Underfitting:** Caracteriza-se por uma baixa perda no treinamento, mas uma alta perda no teste. Isso ocorre porque o modelo possui complexidade excessiva, sendo mais flexível do que o necessário.
B) **Overfitting:** Caracteriza-se por um modelo que não consegue classificar corretamente a maioria dos exemplos do conjunto de treinamento. Isso é causado por excesso de ruído (*noise*) inerente à geração dos dados, que é insuperável.
C) **Underfitting:** A perda no treinamento permanece alta porque o modelo não consegue descrever a verdadeira relação subjacente entre entrada e saída, resultando em um alto **Bias** (vício).
D) **Overfitting:** A perda no teste diminui continuamente, mas a perda no treinamento atinge um patamar máximo e começa a aumentar. Este padrão é conhecido como *double descent*.
E) **Overfitting:** O modelo aprende características específicas aos exemplos de treinamento, apresentando desempenho excelente no treinamento, mas desempenho insatisfatório em novas entradas, o que está ligado a uma alta **Variância**.

Questão 8

Explique os conceitos de campo receptivo (local receptive field) e compartilhamento de pesos (weight sharing) em CNNs. Como estes conceitos estão conectados?

Questão 9

Explique o propósito da camada de max pooling em CNNs. Como ela afeta os feature maps?

Questão 10

Redes Neurais Convolucionais (ConvNets), como a arquitetura LeNet, estabeleceram o *blueprint* para o processamento de imagens, dividindo o modelo em um **extrator de características** (camadas convolucionais e de *pooling*) e um **classificador** (camadas totalmente conectadas).

Utilizando a biblioteca PyTorch, crie um esboço de código Python para uma classe que implemente uma rede ConvNet de arquitetura LeNet-like, respeitando as seguintes especificações:

1. **Entrada:** Imagens coloridas de 32×32 pixels (3 canais).

2. **Estrutura do Extrator de Características (Camadas Conv/Pooling):** Mantenha a estrutura sequencial de convolução (kernel 5×5), seguida por ativação ReLU e *max pooling* (2×2), repetida duas vezes, reduzindo progressivamente a dimensionalidade espacial.
3. **Estrutura do Classificador (Camadas Densas):** Considere duas camadas totalmente conectadas intermediárias, seguidas pela camada de saída final.
4. **Saída:** Um vetor de logits (scores brutos) para a **classificação em 5 classes** distintas.

Você deve incluir a definição da classe (usando `nn.Module` do PyTorch) e o método `forward` que detalha a passagem dos dados.

```
import torch
import torch.nn as nn
import torch.nn.functional as F

# A estrutura geral de uma rede neural é tipicamente definida usando nn.Module,
# especificando a ordem dos componentes e os pesos treináveis.

class LeNet5Classes(nn.Module):
    def __init__(self):
        super(LeNet5Classes, self).__init__()

        # === 1. Extrator de Características (Feature Extractor) ===

        # Imagem de Entrada: 3 canais (RGB) x 32 x 32

        # C1: Camada Convolucional 1 (k=5x5)
        # 3 canais de entrada -> 6 canais de saída
        # Saída espacial: (32 - 5) + 1 = 28. -> 6 x 28 x 28
        self.conv1 = nn.Conv2d(in_channels=_____, out_channels=_____, kernel_size=_____)

        # P1: Max Pooling (k=2, s=2)
        # Saída espacial: 28 / 2 = 14. -> 6 x 14 x 14

        # C2: Camada Convolucional 2 (k=5x5)
        # 6 canais de entrada -> 16 canais de saída
        # Saída espacial: (14 - 5) + 1 = 10. -> 16 x 10 x 10
        self.conv2 = nn.Conv2d(in_channels=_____, out_channels=_____, kernel_size=_____)

        # P2: Max Pooling (k=2, s=2)
        # Saída espacial: 10 / 2 = 5. -> 16 x 5 x 5

        # === 2. Classificador (Fully Connected Layers - MLP) ===

        # FC1: Camada Densa 1. Entrada 400 (16*5*5) -> Saída 120
        self.fc1 = nn.Linear(_____, 120)

        # FC2: Camada Densa 2. Entrada 120 -> Saída 84
        self.fc2 = nn.Linear(120, _____)

        # FC3: Camada de Saída. Entrada 84 -> Saída 5 (para 5 classes)
        self.fc3 = nn.Linear(84, _____)

    def forward(self, x):
        # 1. Primeira Camada: Conv + ReLU
        x = ________________________
        x = F.max_pool2d(x, kernel_size=2, stride=2)
```

```
        # 2. Segunda Camada Conv + ReLU
        x = _________________________
        x = F.max_pool2d(x, kernel_size=2, stride=2)

        # 3. Transforma o tensor 4D (batch_size, channels, height, width) em 2D
        # (batch_size, features)
        x = _________________________

        # 4. Camadas Totalmente Conectadas (MLP)
        x = _________________________
        x = F.relu(self.fc2(x))

        # 5. Saída: Camada final de logits (5 neurônios)
        output = self.fc3(x)

        # Para classificação, a saída final são os logits,
        # que seriam usados com a função de perda de cross-entropy.
        return output

# net = LeNet5Classes()
# dummy_input = torch.randn(1, 3, 32, 32)
# output_logits = net(dummy_input)
```

Questão 11

Quantos bits são necessários para representar uma imagem 256 x 256 com 32 tons de cinza? Descreva o passo-a-passo para os seus cálculos.

Questão 12

Descreva o que é e como são computadas as diferentes vizinhanças de um pixel em uma imagem.

Questão 13

Explique o processo de convolução em imagens em tons de cinza (grayscale). Além disso, ilustre o processo de convolução para uma imagem 9 x 9, com um filtro 3x3.

Questão 14

Explique a diferença entre transformações ponto-a-ponto e filtragem espacial. Em quais aplicações você utilizaria cada uma delas?

Questão 15

Sejam $L_1, L_2 : \mathbb{R}^m \to \mathbb{R}^m$ duas funções de perda (loss), tais que $L_1(\mathbf{y}, \mathbf{y}') \leq L_2(\mathbf{y}, \mathbf{y}')$ para todos $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^m$.

Seja $(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, N$ um conjunto de dados de treinamento.

Seja ainda $\hat{f}$ um modelo de rede neural que minimiza a função de perda L_2 no conjunto de treinamento.

Neste contexto, $\hat{f}$ também minimiza a perda L_1 para os dados de treinamento? Justifique sua resposta.

Questão 16

Como a escolha da função $T(\cdot)$, e.g., linear, gama, logarítmica, **afeta a percepção visual** de contraste e detalhes finos na imagem resultante?
