REDES CONVOLUCIONAIS

LISTA DE EXERCÍCIOS

Redes Convolucionais

I) A rede convolucional LeNET-5 ilustrada na Figura 1 foi desenvolvida por Yann LeCun e publicada em 1989. Sobre a LeNET-5, responda as perguntas abaixo.

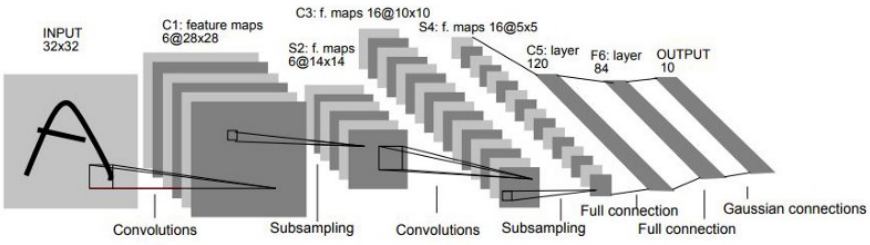

Figure 1: LeNET-5

a) Qual o motivo do número 5 no nome dessa rede?

b) Quais os tamanhos das convoluções usadas?

c) Qual o tamanho do campo receptivo de uma unidade dos mapas de ativação 14x14 gerados após a camada de *average pooling* (*subsampling*)?

d) Quantos parâmetros treináveis tem essa rede?

e) A maior parte dos parâmetros treináveis está localizada nas camadas convolucionais ou nas totalmente conectadas?

II) O que é o *stride* em uma camada convolucional?

III) O que é o *dilation* em uma camada convolucional?

IV) O que é uma convolução transposta?

V) Por que dizemos que camadas de agrupamento do tipo *Max Pooling* e *Average Pooling* fazem *subsampling* dos mapas de ativação?

VI) Por que usualmente omitimos o número de canais em uma convolução 2D?

VII) Considerando uma convolução 2D 3x3 que conecta dois mapas de ativação com tamanho 224x224 com C_1 e C_2 canais respectivamente. Como você poderia inicializar os pesos dessa camada usando a inicialização He?

Conexões Residuais

VIII) Explique porque os dois ramos de um bloco residual não são correlacionados. Mostre que a variância da soma de duas variáveis aleatórias não correlacionadas é igual a soma das variâncias.

IX) Por que precisamos usar *batch norm* quando usamos *skip connections?*

X) Quantos parâmetros treináveis uma camada de *batch norm* possui?

XI) ☠ O grafo computacional do *batch norm* é dado ela Figura 2. Nessa figura, os circulos marcam cada uma das funções que são executadas durante o processo de um forward que aplica *batch norm*. Tendo em vista esse contexto, faça o que é solicitado nas questões abaixo:

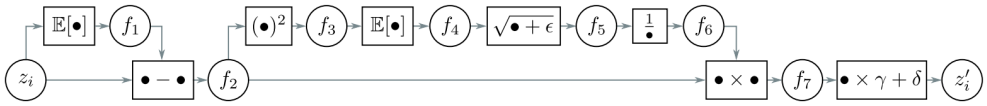

Figure 2: Grafo Computacional Batch Norm

As equações abaixo detalham o grafo computacional:

$$f_1 = \mathbb{E}[z_i] \qquad f_{2i} = x_i - f_1 \qquad f_{3i} = f_{2i}^2 \qquad f_4 = \mathbb{E}[f_{3i}]$$

$$f_5 = \sqrt{f_4 + \epsilon} \qquad f_6 = \frac{1}{f_5} \qquad f_{7i} = f_{2i} \times f_6 \qquad z_i' = f_{7i} \times \gamma + \delta$$

a) Escreva código em Python para implementar o *forward pass*.

b) Descubra as derivadas de cada nodo do grafo.

c) Crie uma expressão que computa $\frac{\partial z_i'}{\partial z_i}$

d) Implemente o *backward pass* em Python.

Tarefas de Visão Computacional

XII) Por que usamos arquiteturas baseadas em *encoder-decoder* para realizar segmentação semântica?

XIII) Explique como as tarefas de localização e detecção de objetos são diferentes.

XIV) Explique como aproveitar uma VGG-16 pré-treinada no ImageNET para realizar classificação de cães e gatos. Sua explicação deve contemplar quais as camadas da VGG-16 você deveria manter e quais você deveria treinar novamente. Suponha que você dispõe de um conjunto de dados relativamente pequeno, com cerca de 250 instâncias para cada classe.

Gabarito

I) a. O número 5 indica o número de camadas com parâmetros treináveis. São 2 convoluções e 3 totalmente conectadas.

b. Na primeira camada convolucional foram convoluções 5x5 com 6 canais de saída; Na segunda foram convoluções 5x5 com 16 canais de saída.

c. Tamanho 6x6. Uma convolução 5x5 tem campo receptivo de tamanho 5x5, se depois dele fizermos um *average pooling* 2x2 teremos *receptive field* de 6x6.

d. Na primeira camada: $5x5x1x6 + 6 = 156$; Na segunda camada: $5x5x6x16 + 16 = 2416$; Na terceira camada $400 * 120 + 120 = 48120$; Na quarta camada $120 * 80 + 80 = 10160$; Na quinta camada $84 * 10 + 10 = 850$. Total 61702.

e. Nas camadas totalmente conectadas.

II) É o quanto o filtro translada entre cada uma das aplicações. Por padrão em convoluções usamos 1, o que significa que o kernel é transladado de uma posição após cada aplicação do filtro.

III) É um parâmetro usado para aumentar o campo receptivo da convolução. Na prática tornamos o kernel esparso completando com zeros.

IV) É uma convolução usada para fazer *upsampling* na imagem. É uma forma de aprender a função de *upsampling*.

V) Pois normalmente usamos essas camadas para diminuir a resolução espacial dos mapas de ativação.

VI) Pois usualmente o número de canais da convolução é igual ao número de canais dos mapas de ativação.

VII) A inicialização He leva em conta o número de parâmetros que contribuem para cada ativação. Como nesse caso estamos falando de C_1x3x3 pesos, podemos inicializar com uma normal com média zero e variância $\frac{2}{9C_1}$

VIII) Como um dos ramos de uma conexão residual passa por uma multiplicação por um vetor de pesos seguido de uma função de ativação não-linear, as correlações lineares são perdidas.

IX) Para não causar explosão das ativações nem dos gradientes devido ao aumento da variância.

X) 2 γ e δ

XI) Ver aqui

XII) Teríamos duas alternativas se não fossemos fazer com *encoder-decoder*: 1 - Usar convoluções sem fazer *downsampling*, o que ficaria inviável devido ao número de parâmetros necessários e quantidade de processamento gasto; 2 - Fazer uma rede convolucional que tem *downsampling* e rodar ela várias vezes em diferentes patches da imagem, o que seria proibitivo também em termos de custo computacional. Para realizar a segmentação da imagem em apenas uma passada, a arquitetura *encoder-decoder* possibilita toda a segmentação ser realizada em uma passada e mantém o número de parâmetros baixo.

XIII) A localização é uma tarefa mais simples que detecção de objetos pois só estamos trabalhando com um objeto de interesse por imagem. Por outro lado, na detecção de objetos não sabemos de antemão quantos objetos estarão presentes na imagem.

XIV) As camadas convolucionais da VGG-16 são responsáveis por extrair características básicas de baixo nível, como bordas e texturas, não necessitam de novo treinamento. Por outro lado, as camadas densas finais estão intimamente relacionadas ao conjunto de dados original (ImageNet). Uma abordagem usual é manter as camadas convolucionais e treinar novamente as camadas densas na extremidade da rede. Isso é um exemplo de *transfer learning* pois estamos aproveitando um aprendizado realizado em um vasto conjunto de dados para ajustar para o nosso problema específico.