Relatório 3

Convolutional Neural Networks
usando *tensorflow*

1th Pedro Vidal Sales
Universidade Federal da Bahia
Tópicos em Computação Visual 3
Professor: Maurício Pamplona

I. Introdução

Esse relátorio explica os parâmetros utilizados para treinar o modelo de rede convolucional para a tarefa de classificar digitos, utilizando a biblioteca *tensorflow*.

II. CNN

A. Divisão entre treino e validação

A base de dados utilizada possui 5000 imagens disponíveis para treino, divididas igualmente em 10 classes, uma para cada digito. A base foi ordenada e foi fixada uma *seed* com valor 1, para que fosse possível recuperar os conjuntos de treino e validação. Após carregar a base, os dados foram permutados aleatoriamente (imagens e labels correspondentes), e depois divididos entre treino e validação. As primeiras 4000 imagens (depois da permutação) foram utilizadas no conjunto de treino, e as outras 1000 imagens foram utilizadas para validação.

B. Treino

Todos os modelos analisados foram treinados por 50 épocas. Cada época corresponde a uma passada por todos os exemplos da base. A cada época o conjunto de validação foi permutado aleatoriamente, para que os mini-batchs de cada época fossem diferentes. Cada mini-batch possui 8 exemplos. O número de passos utilizado foi o número de exemplos do conjunto de treino dividido pelo tamanho do mini-batch, para garantir que cada exemplo da base só seria utilizado uma vez por época. Os valores dos pesos e bias foram atualizados com base no otimizador Adam e na taxa de aprendizado. Para a taxa de aprendizado, foram testados os valores 0.005, 0.0005 e 0.00005, para uma mesma arquitetura, e a taxa que obteve melhores resultados foi 0.0005, por isso ela foi a utilizada para o treinamento das outras arquiteturas. Taxas menores demoravam para alcançar bons resultados, e taxas maiores não alcançavam bons resultados. A função de ativação utilizada nas camadas convolucionais e nas camadas *fully connected* foi a função ReLU. A função de ativação utilizada para calcular as probabilidades de cada classe foi a função sigmoid. Os pesos e bias foram inicializados utilizando o inicializador *global_variables_initializer* da própria biblioteca.

C. Resultados

Os resultados obtidos e as arquiteturas utilizadas estão descritos na tabela I. Primeiro foram testadas redes puramente convolucionais, e depois foram adicionadas camadas densas ao final da rede, o que aumentou a acurácia.

Tamanho da Imagem	Número de filtros por camada de convolução			Número de nós por camada densa			Learning Rate	Resultado
	1ª	2ª	3ª	1ª	2ª	3ª		
77x71	36	64	128	10	-	-	5e-3	96.9%
77x71	36	64	128	10	-	-	5e-4	97.2%
77x71	36	64	128	10	-	-	5e-5	95.5%
64x64	36	64	128	10	-	-	5e-4	97.3%
64x64	36	64	128	128	10	-	5e-4	97.5%
64x64	16	32	128	128	256	10	5e-4	97.6%

Tabela I
Comparação dos resultados obtidos