Trabalho Intermediário de EEE928 - Extração de Caracteríticas

Rafael Fernandes Gonçalves da Silva
Departamento de Engenharia Elétrica
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
rafaelfgs@ufmg.br

***Resumo*—O trabalho consiste em propor novas formas de realizar a extração de características em bases de dados de imagens. As características foram obtidas através da camada totalmente conectada de uma Rede Neural Convolucional (*CNN*) . Para treinamento da rede, foi utilizado o método *backpropagation* utilizando o erro gerado entre os valores de características obtidos e os valores determinados como ideais. Como base de dados, foram utilizados a *MNIST* e a *CIFAR10*, onde foram obtidos valores de acurácia satisfatórios.**

***Palavras-chave*—Rede Neural Convolucional, Extração de Características, *backpropagation*, Máquinas de Vetores Suporte**

I. Introdução

As redes neurais convolucionais também, conhecidas como *CNN's* (*Convolutional Neural Networks*), são amplamente utilzadas como formas de resoluções de diversos problemas computacionais. Dentre eles pode-se citar o reconhecimento de expressões faciais, como realizado em [1]. Em [2], utilizou-se redes convolucionais para classificar textos biomédicos, mais especificamente em artigos de pesquisa e documentos clínicos. Pode também serem utilizadas em treinamentos não supervisionados, como descrito em [3]. *CNN's* são aplicadas em [4] para aprender atividades funcionais de sequências de DNA. Em [5], utilizou-se dessas redes para tarefas de reconhecimento de fala contínua de grande vocabulário (*LVCSR*). As redes neurais convolucionais também podem ser projetadas para segmentar imagens de tecidos cerebrais, conforme realizado em [6].

A extração de características pode representar um método interessante quando visto de forma computacional. Através da extração, conforme explicado em [7], pode-se reduzir a quantidade de dados, aumentar o entendimento sobre a configuração dos dados, melhorar o desempenho da classificação e outras coisas mais. A extração de características do presente artigo consiste em retirar informações da camada totalmente conectada de uma rede e utilizá-las para o treinamento dessa rede.

Vários são os métodos para treinamento de redes neurais convolucionais. O *backpropagation* representa um dos métodos mais antigos de treinamento, já sendo utilizado há algumas décadas, que consiste em propagar, de forma reversa na rede, derivadas para seus parâmetros. Porém, de acordo com [8], esse método é ainda popular devido sua simplicidade conceitual e computacional, geralmente funcionando para vários tipos de base de entrada. Além disso, conforme [9], a derivação passo-a-passo dos parâmetros é de facil entendimento, ajudando os iniciantes no assunto.

Dentre os métodos de classificação, Máquinas de Vetores Suporte (*SVM's*) são amplamente utilizadas atualmente. De acordo com [10], podem ser usadas de forma linear ou não linear, através de transformadas dos dados de entrada, utilizando margens de separação entre as classes. Esse método possui um desempenho significativo, sendo utilizado na mais diversas áreas. Em [11], [12] e [13] pode-se observar aplicações nas respectivas áreas financeira, química e hidrológica.

A base de dados utilizada em um treinamento corresponde outro assunto que possui uma grande variedade de perfis. As bases *MNIST* e *CIFAR10* são duas das mais comumente utilizadas atualmente. A primeira é mais simples e apresenta imagens dos dígitos de 0 à 9 escritos à mão. Possui apenas uma camada de cor (em escala de cinza), tendo 60000 amostras de treino e 10000 amostras de teste. Alguns dos dados da base *MNIST* estão representados na Figura 1(a). A base de dados *CIFAR10* é um pouco mais complexa, possui 50000 amostras de treino e 10000 amostras de teste. Esta apresenta imagens coloridas (camadas de cores RGB), com valores de classes entre 0 e 9 representando os respectivos objetos: *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship* e *truck*. Parte dos dados dessa base está representada na Figura 1(b).

Nesse artigo, será apresentada a utilização de uma rede neural convolucional treinada pelo método *backpropagation* com as bases de dados *MNIST* e *CIFAR10*. O erro calculado na saída para iniciar a propagação das derivadas dos parâmetros foi obtido através da camada totalmente conectada, não sendo necessário um classificador dos dados durante o treinamento. Após o treinamento, os dados de saída são classificados através de *SVM's* e é realizada a predição dos dados de teste.

(a) (b)

Figura 1. Parte da base de dados *MNIST* (a) e *CIFAR10* (b).

II. METODOLOGIA

O objetivo desse trabalho consiste em dois treinamentos. O primeiro, de uma rede neural convolucional, utilizando a camada totalmente conectada para o cálculo do erro. O segundo é o treinamento do classificador, aplicado na rede já treinada, para realizar a predição das classes. Os passos utilizados nesse treinamento estão demonstrados nas subseções a seguir.

Foi proposto no trabalho a utilização da base de dados *MNIST*. Os valores apresentados nessa seção correspondem àqueles obtidos para essa base. Os valores para outras bases estarão apresentados assim quando for necessário. Outra particularidade está na linguagem utilizada. Através do *software RStudio*, versão 1.2.1335, foi implementado um código na linguagem *R*.

A. Rede Neural Convolucional

A estrutura escolhida para a rede neural convolucional foi semelhante àquela apresentada por [9]. Esta foi dividida em seis diferentes camadas e está representada na Figura 2. Inicialmente, uma rede é criada seguindo o padrão utilizado por [9]. Os valores das máscaras utilizadas na convolução (representando os pesos da rede), são iniciados utilizando uma distribuição normal e dependentes do tamanho da máscara e da quantidade de camadas da entrada e da saída, conforme a equação (1). Os limiares responsáveis pela ativação da rede são inicializados como nulos.

$$k_{p,q} = U\left(\pm\sqrt{\frac{6}{(inputsize + outputsize) \times kernel^2}}\right) \quad (1)$$

A primeira camada da rede neural representa a entrada da rede. Nessa camada não é feita modificação alguma e sua saída é a própria imagem de entrada, com dimensão igual à 28x28x1.

A segunda camada representa uma camada de convolução. Neste ponto, passa-se seis máscaras de tamanho 5x5 com passo igual à 1 sobre a camada de saída anterior, representada pela imagem de entrada. Em seguida, a resposta é somada ao limiar e a ativação dessa camada ocorre de acordo com um sigmoide aplicado sobre o resultado dessa soma. Essa operação pode ser representada pelas equações (2) e (3):

$$z_q = \sum_{p=1}^{n} (S_p * k_{p,q}) + b_q \quad (2)$$

$$\sigma = \frac{1}{1 + e^{-z}} \quad (3)$$

onde S_p é a *p-ésima* camada da imagem de entrada, n representa o número de camadas da imagem de entrada, $k_{p,q}$ é a máscara relacionada à *p-ésima* camada da imagem de entrada e à *q-ésima* camada da imagem de saída e b_q é o limiar da *q-ésima* camada da imagem de saída. De acordo com o tamanho e a quantidade das máscaras utilizado, a convolução irá produzir uma imagem de dimensão 24x24x6, representando a saída dessa camada.

A terceira camada representa um escalonamento de imagem, denominado *pooling*. Este processo consiste em reduzir uma imagem através de um fator de escala. Neste trabalho, o tipo de escalonamento utilizado foi o *average pooling*, com um fator de escala igual à 2, reduzindo pela metade o número de dados da imagem. A saída então será representada por um *array* de dimensão 14x14x6. Pode-se notar que nesse método não é feito alterações na quantidade de camadas da imagem.

Na quarta camada há outra operação de convolução, ocorrendo de forma similar à da segunda camada, porém com doze diferentes máscaras. Foi criado um esquema para representar essa operação, mostrado na Figura 3. No esquema, é possivel observar algumas das camadas da imagem de entrada. Cada uma dessas camadas são convolucionadas com as doze máscaras, representadas por $k_{p,q}$, de acordo com (2) e, após serem somadas aos doze diferentes limiares, representados por b_q, são ativados através de um sigmoide σ calculado conforme (3). A saída para essa camada será de dimensão 8x8x12.

A quinta camada é representada por um outro escalonamento de imagem com fator igual à 2, similar à terceira camada. Logo chega-se a valores de saída com dimensão igual à 4x4x12.

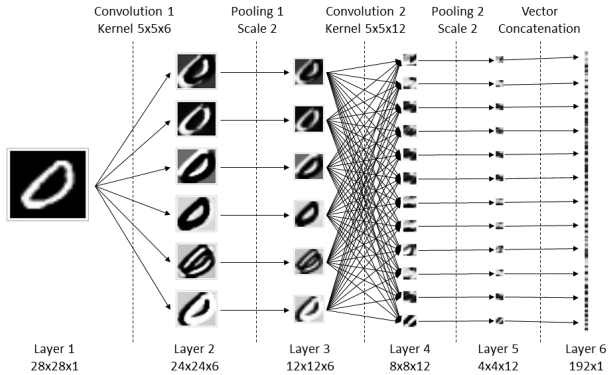

Figura 2. Estrutura da Rede Neural Convolucional utilizada aplicada em uma imagem da classe 0 da base *MNIST*.

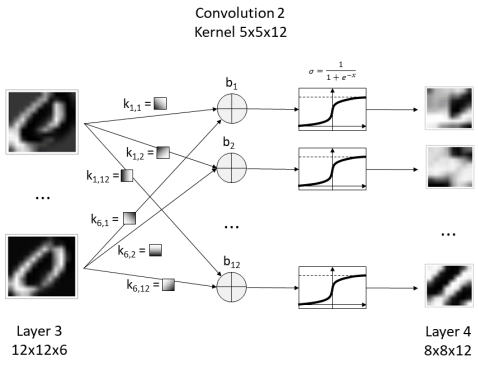

Figura 3. Exemplo da segunda convolução realizada na quarta camada da rede para uma imagem da classe 0 da base *MNIST*.

Por fim, na última camada é realizada uma transformação da saída da última camada em um vetor. Nesse ponto, foi utilizada a função $as.vector$, nativa da linguagem *R*, a qual converte uma variável com parâmetros distribuídos em um vetor não distribuido. Como a ordem dos valores do vetor de saída mostrou-se irrelevante, a utilização dessa função mostrou-se eficiente, principalmente no seu tempo de execução. O tamanho do vetor de saída pode ser representado pelo produto das dimensões da entrada dessa camada, sendo igual à 128x1.

B. Treinamento da rede

Dada a rede neural convolucional criada, seu treinamento pode ser representado por um ciclo, que pode ser dividido em cinco partes, mostrado na Figura 4. Para um melhor desempenho computacional, as imagens de entrada são divididas em pacotes de índices aleatórios que são lançados na rede. Cada entrada de um pacote de imagens representa uma iteração, onde os pesos e os limiares das camadas são atualizados.

A primeira refere-se ao *feedfoward*, onde um pacote de imagens é definido como entrada. De acordo com os parâmetros atuais da rede, são determinadas de forma sequencial as saídas correspondentes à cada camada. As características utilizadas nos passos seguintes do ciclo são aquelas encontradas na sáida da última camada da rede.

Na segunda parte do treinamento são obtidas as características ideais para as imagens de entrada. Para encontrar essa matriz, foi realizada uma sequência de passos. Primeiramente, utilizando a saída da última camada e os valores de suas respectivas classes, foi encontrada a média das características de cada classe. A Tabela I demonstra a média encontrada para dez características em um dado instante do treinamento. Em seguida, foi encontrada uma matriz que representa um histórico do andamento das classes, iniciada com valores nulos. A cada iteração, para uma característica qualquer, os valores encontrados na matriz anterior que estiverem abaixo da média são adicionados como -1 e caso estiverem abaixo da média são adicionados como 1 à essa matriz de histórico, mostrada na Tabela II. Utilizando essas duas matrizes, foi estabelecido que a segunda irá prevalecer caso, para uma característica qualquer, o histórico de alguma classe for maior que a média somada ao dobro do desvio padrão ou menor que a média subtraída do dobro do desvio padrão, resultando respectivamente em 1 ou 0 na matriz final. Para valores intermediários do histórico, a matriz final será dada pela matriz das médias, 1 para valores acima da média e 0, caso contrário. Por fim, é verificada a separação entre as colunas dessa matriz final, utilizando a média dos quadrados das diferenças entre elas, apresentada na Tabela III. Caso algum desses valores for superior a 0.7 (valor determinado empiricamente, representando uma diferença de 30% entre as classes), é realizada a separação entre as classes correspondentes à esse valor. A separação é determinada utilizando a matriz final e a matriz de médias da Tabela I, onde são trocados alguns dos valores semelhantes na matriz final entre as classes não separadas, observando a diferença entre eles na matriz de média (a quantidade de valores a serem trocados é proporcional à semelhança entre as classes encontrada na matriz da Tabela III). Essa tentativa de separação é realizada quatro vezes, chagando-se a esse limite, ou ocorrendo algum erro no caminho, a tentativa de separação é finalizada e a matriz final continua com os valores atuais. Nesse ponto a matriz final é determinada como ideal para cada classe e utilizando os valores das classes do pacote atual, a matriz final é convertida para vetores de características ideais para a entrada dada.

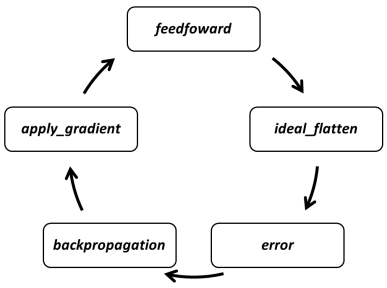

Figura 4. Ciclo utilizado no treinamento da rede.

Na terceira parte do treinamento, são gerados os erros através da diferença entre os vetores encontrados na iteração atual e o vetor ideal encontrado na parte anterior. Através desse erro é encontrado o valor de perda atual, correspondente à média das somas das colunas dos erros ao quadrado. Este cálculo está apresentado em (4), onde i e j representam as linhas e as colunas do erro E, n é o número total de linhas de E e a média é aplicada em um vetor de colunas. Com esse valor, pode-se ter uma noção da quantidade de características que não atingiram se valor ideal.

$$L = mean\left[\sum_{i=1}^{n} E_{i,j}^{2}\right] \tag{4}$$

Na quarta parte do treinamento, através dos valores de erro encontrados na seção anterior, é utilizado o método *backpropagation*, onde percorre-se a rede neural de trás pra frente e são determinadas as derivadas das saídas de cada camada, de forma a minimizar o erro. Em seguida, a rede é novamente percorrida, dessa vez no seu sentido normal, e são encontradas as derivadas dos pesos e dos limiares que correspondem aos valores das derivadas das saídas.

Na quinta e última parte do ciclo de treinamento, são aplicados os gradientes dos pesos de dos limiares. Para isso, seus valores atuais são somados aos valores de suas respectivas derivadas multiplicados pelo fator de treinamento determinado antes do treino. A partir daí, com os pesos e limiares atualizados, reinicia-se o ciclo aplicando novamente dados de entrada.

C. Treinamento do classificador

Depois do treinamento da rede, foi realizado o treinamento do classificador através dos valores das características, representados pela saída da última camada da rede. Foi utilizado Máquinas de Vetores Suporte (*SVM's*) como classificador das classes para as dadas características.

Tabela I
MÉDIA DAS DEZ PRIMEIRAS CARACTERÍSTICAS DE CADA CLASSE PARA UM DADO INSTANTE DO TREINAMENTO.

4	2	8	7	1	3	5	6	9	0
0.66	0.27	0.65	0.71	0.25	0.29	0.67	0.37	0.84	0.76
0.83	0.18	0.29	0.13	0.17	0.27	0.55	0.54	0.67	0.58
0.37	0.76	0.37	0.15	0.15	0.13	0.27	0.77	0.15	0.76
0.06	0.45	0.59	0.38	0.22	0.70	0.56	0.24	0.21	0.77
0.82	0.27	0.78	0.72	0.37	0.45	0.87	0.69	0.94	0.76
0.84	0.25	0.64	0.08	0.41	0.51	0.66	0.90	0.66	0.33
0.34	0.83	0.67	0.15	0.59	0.21	0.39	0.91	0.13	0.52
0.09	0.41	0.66	0.47	0.67	0.78	0.68	0.28	0.29	0.78
0.81	0.24	0.68	0.74	0.75	0.53	0.70	0.60	0.84	0.62
0.89	0.47	0.63	0.13	0.79	0.67	0.52	0.82	0.59	0.13

Tabela II
HISTÓRICO DAS DEZ PRIMEIRAS CARACTERÍSTICAS PARA UM DADO INSTANTE DO TREINAMENTO.

4	2	8	7	1	3	5	6	9	0
4819	-7201	-6805	6025	7023	4969	-5761	4237	6873	-6827
3491	-7201	-5459	7101	6745	5593	5127	-4803	-7037	-5425
-5367	-7013	-6265	-1895	-6091	7119	6735	-3007	-7031	7065
6221	-6503	6969	-7203	-7113	7049	-6813	5483	-5483	2507
6521	-7165	-6781	5385	7063	5795	1493	6177	6301	-7113
5325	-6907	-169	7081	5913	-6117	7081	4315	-7175	-5913
-5345	5555	-6951	-5301	-6949	3921	7093	5433	-7023	7109
5831	6111	6967	-7193	-7007	6857	-7085	6939	-3177	-3545
1243	6425	-4499	2217	6357	247	-2067	4205	6519	-7095
-3073	7099	6311	6907	3583	-7183	6933	4547	-7159	-1195

Tabela III
SEMELHANÇA ENTRE AS CLASSES OBSERVADA ANTES DA SEPARAÇÃO PARA UM DADO INSTANTE DE TREINO.

	4	2	8	7	1	2	5	6	9	0
4	1.00	0.33	0.73	0.53	0.66	0.46	0.52	0.59	0.43	0.26
2	0.33	1.00	0.33	0.53	0.47	0.45	0.46	0.18	0.59	0.52
8	0.73	0.33	1.00	0.33	0.51	0.52	0.32	0.68	0.54	0.40
7	0.53	0.53	0.33	1.00	0.70	0.22	0.71	0.41	0.40	0.41
1	0.66	0.47	0.51	0.70	1.00	0.38	0.46	0.47	0.67	0.14
3	0.46	0.45	0.52	0.22	0.38	1.00	0.33	0.46	0.56	0.58
5	0.52	0.46	0.32	0.71	0.46	0.33	1.00	0.54	0.15	0.56
6	0.59	0.18	0.68	0.41	0.47	0.46	0.54	1.00	0.38	0.54
9	0.43	0.59	0.54	0.40	0.67	0.56	0.15	0.38	1.00	0.34
0	0.26	0.52	0.40	0.41	0.14	0.58	0.56	0.54	0.34	1.00

Tabela IV
VALORES IDEAIS PARA AS CARACTERÍSTICAS SEPARÁVEIS DE CADA CLASSE PARA UM DADO INSTANTE DE TREINO.

4	2	8	7	1	3	5	6	9	0
1	0	0	1	1	1	0	1	1	0
1	0	0	1	1	1	1	0	0	0
0	0	0	0	0	1	1	1	0	1
1	0	1	0	0	1	0	1	0	1
1	0	0	1	1	1	1	1	1	0
1	0	1	1	1	0	1	1	0	0
0	1	0	0	0	1	1	1	0	1
1	1	1	0	0	1	0	1	0	0
1	1	0	1	1	0	0	1	1	0
0	1	1	1	1	0	1	1	0	0

Utilizando parte do treinamanto da rede neural convolucional (*feedfoward*), foram lançados na rede todos dados de treino da base *MNIST*. Na última camada, a saída resultante foi uma matriz 192x60000, representando as 60000 características de cada entrada, associada a sua respectiva classe.

Através da função svm do *R*, presente na biblioteca $e1071$, utilizando a transposta da matriz das características e as respectivas classes, obteve-se um modelo do classificador das classes para a saída da rede neural convolucional.

III. RESULTADOS E DISCUSSÃO

Para o treinamento da rede, foram definidos padrões para alguns parâmetros: utilização de todos os dados de treino, uma taxa de aprendizagem de 0.2 e pacotes com tamanho fixo de 50. Para o treinamento do classificador, foi utilizado um *kernel* do tipo radial e um custo de violação de restrições igual à 10.

Os resultados desse trabalho foram obtidos em um Notebook Acer Aspire V3-571, utilizando o Windows 10 Home 64 bits, com um processador Intel® Core™ i7-3632QM CPU @ 2.20GHz, memória RAM 8,00GB 1333MHz e driver de vídeo Intel® HD Graphics 4000.

A. Base de dados MNIST

Para a base de dados *MNIST*, foram utilizadas seis épocas de treinamento da rede, tendo uma duração de 3 horas e 4 minutos. Para pacotes de tamanho 50 e uma base de treino com 60000 imagens, o treino foi resultante de 7200 iterações. Os valores calculados de perda, através de (4), para cada iteração estão apresentados na Figura 5.

Pode-se observar que princípio do treinamento está ocorrendo como esperado, visto que a função de perda diminui a cada iteração. O ruído encontrado no valores de perda ocorre devido ao tamanho reduzido dos pacotes. Aumentar essa quantidade provocaria um menor ruído, mas reduziria o número de iterações e, consequentemente, as aplicações do gradiente.

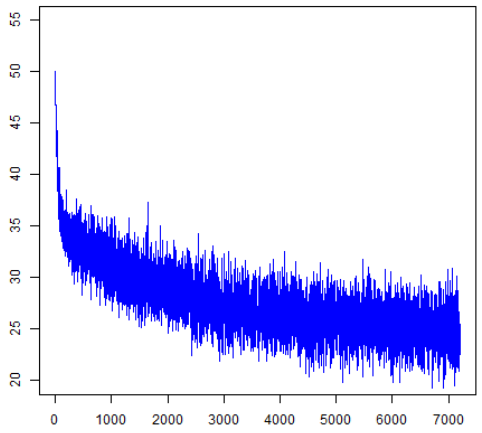

Figura 5. Valores de perda encontrados durante seis épocas de treinamento da base *MNIST* com pacotes de tamanho 50.

Tabela V
VALORES PREDITOS (COLUNAS) PARA CADA CLASSE DO *MINST* (LINHAS).

	0	1	2	3	4	5	6	7	8	9
0	99.1%	0.0%	0.0%	0.0%	0.1%	0.2%	0.3%	0.1%	0.2%	0.0%
1	0.0%	99.6%	0.2%	0.0%	0.0%	0.0%	0.1%	0.1%	0.0%	0.0%
2	0.1%	0.0%	98.8%	0.3%	0.1%	0.0%	0.2%	0.4%	0.1%	0.0%
3	0.0%	0.0%	0.3%	98.1%	0.0%	0.7%	0.0%	0.4%	0.5%	0.0%
4	0.1%	0.0%	0.1%	0.0%	99.0%	0.0%	0.3%	0.0%	0.1%	0.4%
5	0.3%	0.1%	0.0%	1.1%	0.0%	98.1%	0.1%	0.1%	0.0%	0.1%
6	0.3%	0.2%	0.1%	0.0%	0.1%	0.4%	98.7%	0.0%	0.1%	0.0%
7	0.0%	0.1%	1.0%	0.3%	0.1%	0.0%	0.0%	98.4%	0.0%	0.1%
8	0.2%	0.2%	0.2%	0.6%	0.1%	0.4%	0.1%	0.1%	97.7%	0.3%
9	0.1%	0.0%	0.0%	0.3%	0.3%	0.4%	0.0%	0.5%	0.2%	98.2%

Tabela VI
VALORES PREDITOS (COLUNAS) PARA CADA CLASSE DO *CIFAR10* (LINHAS).

	aiplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
aiplane	56.0%	4.5%	4.0%	3.0%	3.2%	2.1%	1.9%	4.4%	14.8%	6.1%
automobile	3.7%	58.0%	2.0%	2.4%	0.9%	1.7%	1.5%	2.7%	11.1%	16.0%
bird	6.9%	2.5%	27.3%	10.3%	18.2%	8.5%	12.4%	8.6%	2.8%	2.5%
cat	3.5%	2.2%	7.6%	31.2%	7.7%	15.3%	17.6%	7.8%	3.6%	3.5%
deer	4.8%	1.6%	6.7%	8.8%	39.9%	6.8%	15.2%	10.6%	3.0%	2.6%
dog	2.1%	2.5%	9.3%	18.9%	10.0%	31.3%	11.8%	9.2%	3.1%	1.8%
frog	0.7%	1.5%	5.5%	9.3%	12.0%	6.3%	58.2%	3.7%	0.9%	1.9%
horse	5.5%	2.4%	4.3%	6.2%	9.5%	8.4%	5.1%	52.9%	1.9%	3.8%
ship	19.3%	7.9%	1.5%	4.2%	0.8%	1.9%	1.5%	2.0%	52.8%	8.1%
truck	6.8%	17.2%	1.8%	3.1%	1.5%	1.7%	3.0%	5.7%	11.8%	47.4%

Para o treinamento classificador *SVM*, foram utilizados os 60000 dados de treino e foi executado com um tempo de 22 minutos e 53 segundos.

Para obter os resultados de acurácia, foram utilizados os 10000 dados de teste da base *MNIST*. Esses dados foram lançados na rede e a saída encontrada na última camada representou, de forma transposta, os dados de entrada para a predição. Esta foi determinada através da função $predict$, nativa da linguagem *R*, e como argumentos foram utilizados os dados citados anteriomente e o modelo encontrado no treinamento do classificador. Realizando a média dos acertos para essas previsões obteve-se um resultado de 98.57%.

Foi também encontrada a acurácia entre as classes, utlizando o mesmo príncipio anteriormente apresentado. Os valores encontrados estão na Tabela V. Através desse resultado, é possível observar algumas peculiaridades. Os números com maior dificuldade de acerto foram o 8 e o 5, com taxas de acerto respectivas à 97.7% e 98.1%. As trocas mais comuns foram a classificação do 5 como 3 e do 7 como 2, com valores superiores à 1%. O melhor resultado observado foi para a classe 1, com 99.6% de acerto.

B. Base de dados CIFAR10

Para demonstrar a funcionalidade do código implementado, foi utilizada a base de dados *CIFAR10* para treinamento da rede neural convolucional, não havendo alteração alguma no código. Para esta, foram utilizadas quatro épocas de treinamento da rede, tendo uma duração de 3 horas e 57 minutos. Neste caso, o treino foi resultante de 4000 iterações. Os valores calculados de perda para cada iteração estão apresentados na Figura 6.

Os 8GB de memória *RAM* do *hardware* disponível não foram suficientes para realizar o treinamento classificador *SVM* com os 50000 dados de treino da base *CIFAR10*, visto que o uso de imagens coloridas aumenta consideravelmente o uso de memória. Para isso, essa quantidade foi reduzida para 20000. Nesse caso, ainda foram observados picos de uso de memória próximos à 5.5GB, logo este pode ser considerado um fator limitante pelo *hardware*. Utilizando então 20000 imagens obtidas de forma aleatória da base *CIFAR10*, observou-se um tempo de treino igual à 19 minutos e 54 segundos.

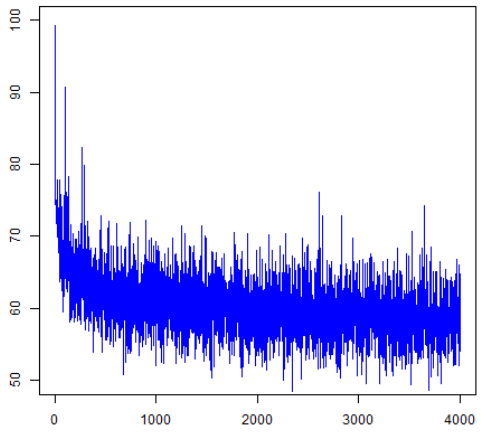

Figura 6. Valores de perda encontrados durante seis épocas de treinamento da base *CIFAR10* com pacotes de tamanho 50.

De forma similar à base *MNIST*, foram obtidos os resultados de acurácia através da função $predict$, utilizando as características encontradas na última camada da rede com os 10000 dados de teste da *CIFAR10*. A média dos acertos apresentada para as previsões foi igual à 45.50%.

Também foram encontrados os valores de acurácias entre as classes, cujos resultados estão na Tabela VI. Através dela, pode-se observar uma menor aproximação entre os valores preditos e os valores reais. A pior classificação ocorreu para a classe *bird*, com 27.3% de acerto, sendo esta confundida com várias outras. A pior confusão ocorreu na classe *ship*, sendo confundida com a classe *airplane* em 19.3% dos casos. Os melhores resultados foram das classes *frog* e *automobile* com 58.2% e 58.0% de acerto.

Mesmo com altos valores de confusão, pode-se perceber que estes ocorreram entre classes mais próximas. As classes que representam animais obtiveram erros maiores quando comparadas entre si, mas erros menores quando comparadas com os meios de transporte. A mesma relação pode ser observada de forma contrária.

IV. Conclusão

Os resultados obtidos nesse trabalho apresentaram um desempenho satisfatório quando comparado a outros resultados de trabalhos similares.

Neste trabalho, o tempo demonstrou-se como um obstáculo para a execução do código, visto que cada época para a base de dados *MNIST* era executada em cerca de 30 minutos e cada época da *CIFAR10* em cerca de 1h. Um maior tempo de treinamento das redes ou a utilização de melhores *hardwares* poderiam apresentar melhores resultados finais. De forma similar, uma maior disponibilidade de memória *RAM* poderia melhorar o treinamento do classificado para a base *CIFAR10*, conforme explicado na subseção III-B

Durante a implementação do código, foram criadas três versões, cada uma com algum avanço em relação à outra. Na primeira, só havia a criação da matriz ideal de características baseada na média, sem garantir a separação, apresentando resultados de acurácia inferiores à 90%. Na segunda versão, foi implementada a separação das colunas da matriz ideal de características, apresentando resultados próximos à 96%. Nese ponto, foi observada uma alta variação nos valores da matriz ideal ao utilizar somente a média como base, logo, na versão atual, foi adicionado o histórico da matriz ideal deixando o treinamento da rede mais estável, visto que a matriz objetivo não apresentava valores com alta oscilação.

A principal proposta para futuros trabalhos é a implementação de uma forma de separação com algum embasamento teórico. A forma de separação atual baseia-se na matriz de médias das características das classes e realiza trocas de parte da matriz ideal das classes que não estão separadas, algo muito próximo à tentativa e erro. Esse método funcionou para as classes utilizadas, porém a utilização de métodos melhores formulados pode aumentar o desempenho final.

Outro fato importante nesse trabalho foi a utilização da média como forma de criar e serapar as colunas da matriz de características ideais. Este método também possui resultados satisfatórios para a base de dados *MNIST* e *CIFAR10*, podendo apresentar algumas limitações para bases com maior complexidade.

Por fim, de forma a aumentar o desempenho dos resultados, pode-se aplicar alguns dos métodos apresentados em [7]: rejeitando ou aceitando características baseado nas suas respectivas relevâncias, ou levando em consideração a dependência entre as características.

Referências

[1] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, "Subject independent facial expression recognition with robust face detection using a convolutional neural network," *Neural Networks*, vol. 16, no. 5-6, pp. 555–559, Jun. 2003. [Online]. Available: https://doi.org/10.1016/S0893-6080(03)00115-1

[2] A. Rios and R. Kavuluru, "Convolutional neural networks for biomedical text classification: application in indexing biomedical articles," Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics, Sep. 2015, pp. 258–267.

[3] J. L. Ba, K. Swersky, S. Fidler, and R. Salakhutdinov, "Predicting deep zero-shot convolutional neural networks using textual descriptions," in *The IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015.

[4] D. R. Kelley, J. Snoek, and J. L. Rinn, "Predicting deep zero-shot convolutional neural networks using textual descriptions," *Neural Networks*, pp. 990–999, May 2016. [Online]. Available: http://www.genome.org/cgi/doi/10.1101/gr.200535.115

[5] T. N. Sainath, B. Kingsbury, G. Saon, H. Soltau, A. rahman Mohamed, G. Dahl, and B. Ramabhadran, "Deep convolutional neural networks for large-scale speech tasks," *Neural Networks*, vol. 64, pp. 39–48, Apr. 2015. [Online]. Available: https://doi.org/10.1016/j.neunet.2014.08.005

[6] W. Zhang, R. Li, H. Deng, L. Wang, W. Lin, S. Ji, and D. Shen, "Deep convolutional neural networks for multi-modality isointense infant brain image segmentation," *NeuroImage*, vol. 108, pp. 214–224, Mar. 2015. [Online]. Available: https://doi.org/10.1016/j.neuroimage.2014.12.061

[7] I. Guyon and A. Elisseeff, "An introduction to feature extraction," in *Feature Extraction*, J. Kacprzyk, Ed. Springer, Berlin, Heidelberg, 2006, vol. 207, pp. 1–25.

[8] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*, ser. Studies in Fuzziness and Soft Computing, J. Kacprzyk, Ed. Springer, Berlin, Heidelberg, 2006, vol. 7700, pp. 9–48.

[9] Z. Zhang, "Derivation of backpropagation in convolutional neural network (cnn)," 2016.

[10] L. Wang, *Support vector machines: theory and applications*. Springer Science & Business Media, 2005, vol. 177.

[11] F. E. H. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," *Omega*, vol. 29, no. 4, pp. 309–317, Aug. 2001. [Online]. Available: https://doi.org/10.1016/S0305-0483(01)00026-3

[12] H. Li, Y. Liang, and Q. Xu, "Support vector machines and its applications in chemistry," *Chemometrics and Intelligent Laboratory Systems*, vol. 95, no. 2, pp. 188–198, Feb. 2009. [Online]. Available: https://doi.org/10.1016/j.chemolab.2008.10.007

[13] S. R. N and P. C. Deka, "Support vector machines and its applications in chemistry," *Applied Soft Computing*, vol. 19, pp. 372–386, Jun. 2014. [Online]. Available: https://doi.org/10.1016/j.asoc.2014.02.002