UFSC - Universidade Federal de Santa Catarina Igor Althoff Vidal
INE 5430 - Inteligência Artificial Fabíola Maria Kretzer
Prof. Mauro

Trabalho 5: A Identificação de Objetos no RX do Aeroporto

1 Introdução

Uma prática muito comum para o reconhecimento de padrões é a utilização de uma rede neural para classificar um conjunto de padrões de entrada. A partir dessa rede já treinada é possível classificar padrões nunca visto anteriormente. Assim o objetivo do trabalho proposto é treinar e testar uma rede neural que classifica peças do vestuário detectadas por um RX do aeroporto. Para isso será utilizado conjuntos de teste, um para treinar e testar a rede e o outro para testar. Mais detalhes da implementação serão explicados na seção de Desenvolvimento.

2 Desenvolvimento

Nessa seção, serão ralizadas explicações do código utilizado e também apresentados diagramas gerados a partir da ferramenta *MatLab*.

2.1 Carregar os módulos e o Conjunto de dados

Para essa etapa do trabalho foram disponibilizados dois arquivos *.m* (padrão do *MatLab*). O primeiro possui o código de uma função, chamada *importFile.m*. Essa função importa os dados dos *.csv* disponibililizados e transforma esses dados numéricos do arquivo de texto em uma matriz. O outro possui o nome de *prepData.m* e utiliza o *importFile* para preparar os dados e importar para o ambiente do *MatLab*.

2.2 Normalização dos dados

Segundo a documentação do *MatLab*, a função *newff* normaliza os dados antes da criação da rede neural. Mas para entender como é realizada a normalização de dados será utilizado a função *mapminmax* antes do treinamento. Essa função pode ser útil para dimensionar as entradas e os destinos para que sempre caiam dentro de um intervalo especificado. A função *mapminmax* dimensiona entradas e destinos para que eles caiam no intervalo [-1,1].

O código de normalização é:

```
[Data_train,PS] = mapminmax(input');
Target_train=TARGET';
Data_test=mapminmax('apply',input_test',PS);
Target_test=TARGET_TEST';
```

2.3 Separação do conjunto de treinamento e de teste

Para esse trabalho foi disponibilizado pelo professor, dois arquivos de dados: *fashion_mnist_test.csv* e *fashion_mnist_train.csv*. Um contém os 60000 dados que são divididos entre teste e validação, sendo 90% e 10% respectivamente. E o outro possui os dados de treinamento.

O código de separação do conjunto de dados é apresentado a seguir:

$$\begin{aligned} &\text{net.divideParam.trainRatio} = 0.9; \\ &\text{net.divideParam.valRatio} = 0.1; \\ &\text{net.divideParam.testRatio} = 0.0; \end{aligned}$$

Como foi disponibilizado um arquivo de dados separado para o teste então não se faz necessário alocar dados de $fashion\_mnist\_train.csv$ para essa atividade.

2.4 Arquitetura e experimentos realizados

Nesta subseção serão apresentados aspectos da arquitetura, dos experimentos, a realização do treinamento e a sua performace, taxa de acerto da classificação e a geração da matriz de confusão. Para demonstrar o domínio do trabalho, foram realizados alguns testes com diferentes configuração. Esses testes utilizam as funções $newff()$ (cria a rede neural conforme alguns parâmetros), $train()$ (treina a rede neural) e $sim()$ (faz a simulação da rede neural) Quatro desses testes serão explicados abaixo.

Teste 1

Essa rede neural é configurada com a camada de entrada com 784 neurônios, duas camadas intermediárias, a primeira usando 30 neurônios e a outra 15, e a camada de saída com 10 neurônios.

O algoritmo de treinamento utilizado foi o RProp (trainrp) que é uma função de treinamento de rede que atualiza valores de ponderação e polarização de acordo com o algoritmo resiliente de retropropagação. Foi configurado para tentar alcançar um erro de 0.000001 e uma taxa de aprendizado de 0.05.

$$\begin{aligned} &\text{net = newff (Data\_train,Target\_train,[30 15],'logsig' 'logsig'} \\ &\text{'purelin','trainrp');} \end{aligned}$$

onde os parâmetros são:

- **Data_train:** é o conjunto de dados para treinamento;
- **Target_train:** matriz numérica contendo o dígito que o conjunto de entradas representa;
- **[30 15]:** número de neurônios nas camada intermediárias;
- **'logsig' e 'logsig':** funções log-simóide de ativação dos neurônios das camadas intermediárias;
- **'purelin':** função de ativação dos neurônios de saída, calcula a saída de uma camada a partir de sua entrada;
- **'trainrp':** algoritmo de treinamento explicado anteriormente.

Em seguida, foi realizado o treinamento da rede usando as funções:

$$\text{net=train(net,Data\_train,Target\_train);}$$

Y=sim(net,Data_train);

A primeira treina a rede neural utilizando os parâmetros passados em *net*. Um desses parâmetros é o número de épocas de treinamento, no qual foram definidos como 500. Enquanto a segunda faz a simulação da rede. A Figura 1 mostra as configurações e o treinamento da rede. Já na Figura 2 é mostrado a performace da rede utilizando os dados de validação e teste.

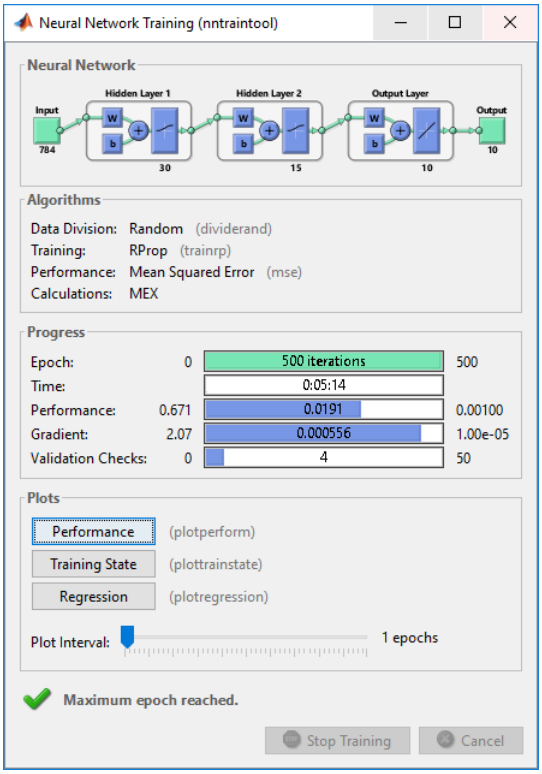

Figura 1: Configurações da rede neural.

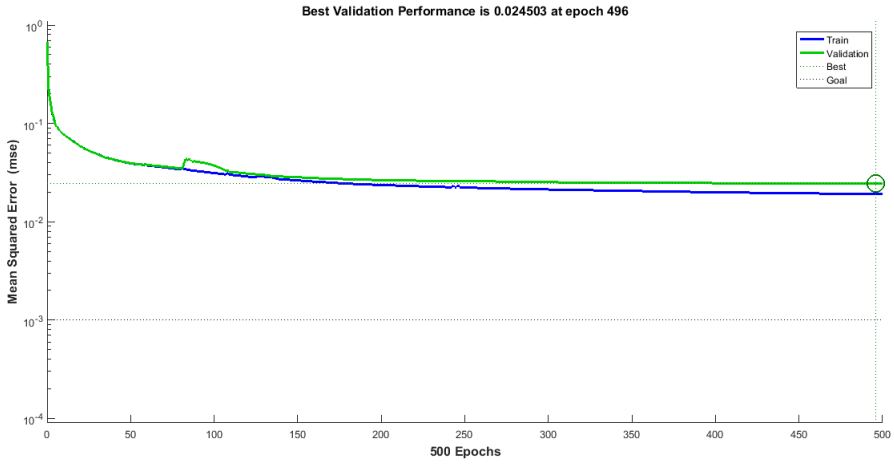

Figura 2: Performace da rede neural.

Após o treinamento é necessário separar os valores obtidos em uma matriz numérica. Assim, é procurado, para cada determinada entrada, a maior nota recebida em relação a todas as saídas

do problema e atribui valor 1 a classe com maior valor (código pode ser visualizado em *rede.m*). Depois a mariz de confusão é gerada. Abaixo são apresentadas a matriz de confusão para os dados de treinamento (com 87,5% de acerto) e de teste (com 84,5% de acerto).

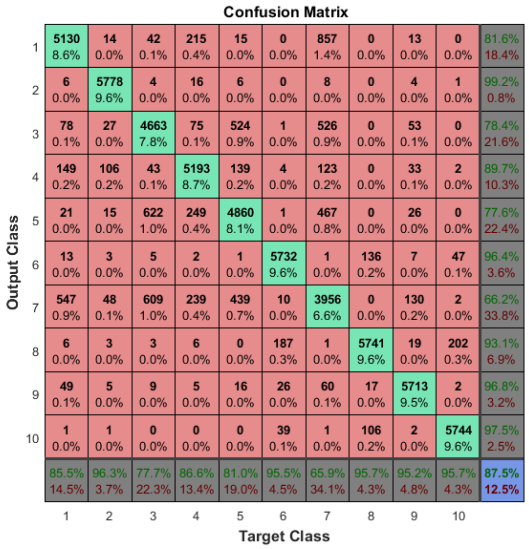

Figura 3: Matriz de confusão com os dados de treino.

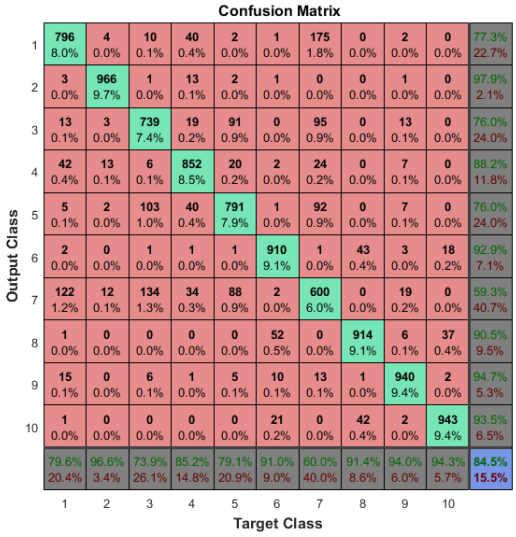

Figura 4: Matriz de confusão com os dados de teste.

Teste 2

Essa rede neural é configurada com a camada de entrada com 784 neurônios, duas camadas intermediárias, a primeira usando 17 neurônios e a outra 11, e a camada de saída com 10 neurônios.

O algoritmo de treinamento utilizado foi o RProp (trainrp) que é uma função de treinamento de rede que atualiza valores de ponderação e polarização de acordo com o algoritmo resiliente de retropropagação. Foi configurado para tentar alcançar um erro de 0.000001 e uma taxa de aprendizado de 0.05.

$$\text{net} = \text{newff (Data\_train,Target\_train,[17 11],'logsig' 'logsig' 'purelin','trainrp');}$$

onde os parâmetros são:

- **Data_train:** é o conjunto de dados para treinamento;
- **Target_train:** matriz numérica contendo o dígito que o conjunto de entradas representa;
- **[17 11]:** número de neurônios nas camada intermediárias;
- **'logsig' e 'logsig':** funções de ativação dos neurônios das camadas intermediárias;
- **'purelin':** função de ativação dos neurônios de saída, calcula a saída de uma camada a partir de sua entrada;
- **'trainrp':** algoritmo de treinamento explicado anteriormente.

Em seguida, foi realizado o treinamento da rede usando as funções:

$$\text{net=train(net,Data\_train,Target\_train);}$$

$$\text{Y=sim(net,Data\_train);}$$

A primeira treina a rede neural utilizando os parâmetros passados em *net*. Um desses parâmetros é o número de épocas de treinamento, no qual foram definidos como 500. Enquanto a segunda faz a simulação da rede. A Figura 5 mostra as configurações e o treinamento da rede. Já na Figura 6 é mostrado a performace da rede utilizando os dados de validação e teste.

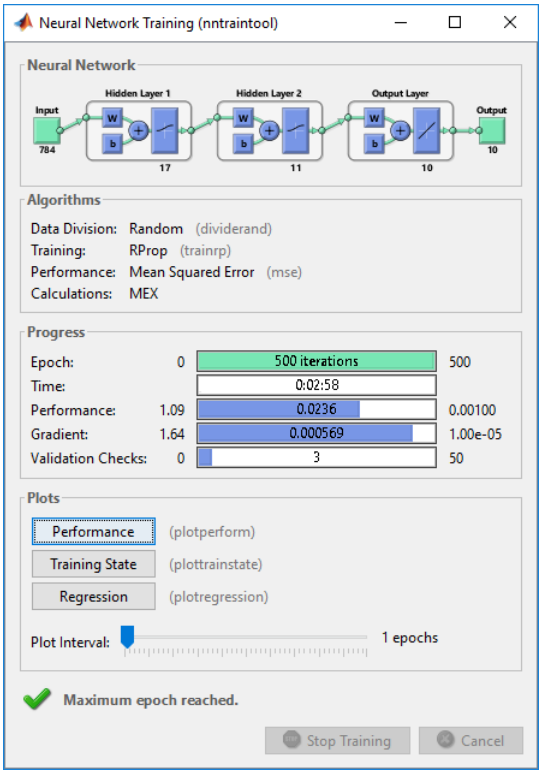

Figura 5: Configurações da rede neural.

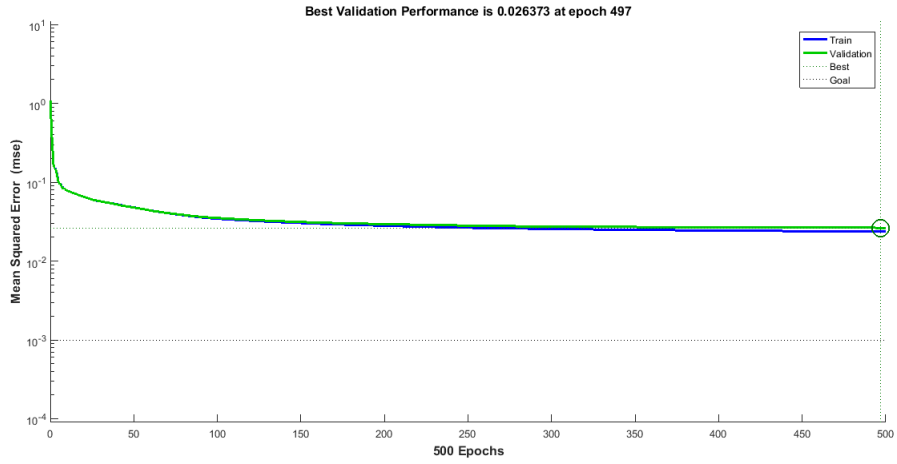

Figura 6: Performace da rede neural.

Após o treinamento é necessário separar os valores obtidos em uma matriz numérica. Assim, é procurado, para cada determinada entrada, a maior nota recebida em relação a todas as saídas do problema e atribui valor 1 a classe com maior valor (código pode ser visualizado em *rede.m*). Depois a mariz de confusão é gerada. Abaixo são apresentadas a matriz de confusão para os dados de treinamento (com 84,7% de acerto) e de teste (com 82,3% de acerto).

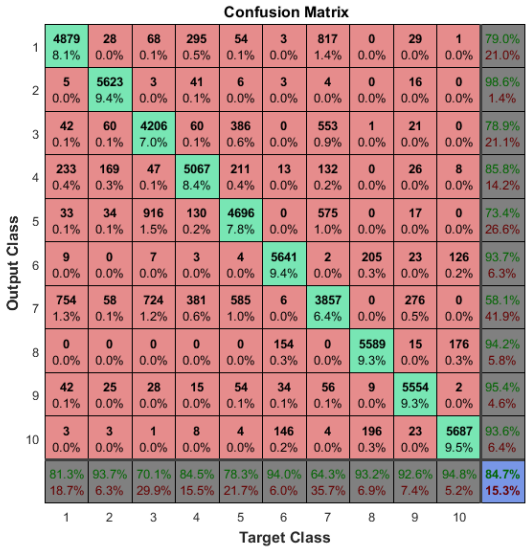

Figura 7: Matriz de confusão com os dados de treino.

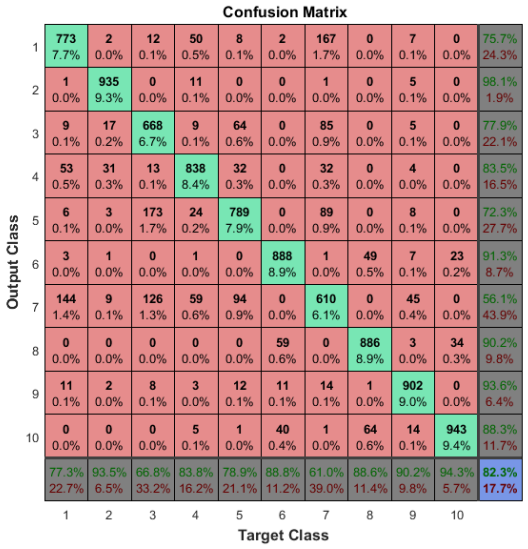

Figura 8: Matriz de confusão com os dados de teste.

Teste 3

Essa rede neural é configurada com a camada de entrada com 784 neurônios, duas camadas intermediárias, a primeira usando 17 neurônios e a outra 11, e a camada de saída com 10 neurônios.

O algoritmo de treinamento utilizado foi o RProp (trainrp) que é uma função de treinamento de rede que atualiza valores de ponderação e polarização de acordo com o algoritmo resiliente de retropropagação. Foi configurado para tentar alcançar um erro de 0.000001 e uma taxa de aprendizado de 0.05.

$$\text{net} = \text{newff (Data\_train,Target\_train,[17 11],'transig' 'transig' 'purelin','trainrp');}$$

onde os parâmetros são:

- **Data_train:** é o conjunto de dados para treinamento;
- **Target_train:** matriz numérica contendo o dígito que o conjunto de entradas representa;
- **[17 11]:** número de neurônios nas camada intermediárias;
- **'transig' e 'transig':** funções tangente-simóide de ativação dos neurônios das camadas intermediárias;
- **'purelin':** função de ativação dos neurônios de saída, calcula a saída de uma camada a partir de sua entrada;
- **'trainrp':** algoritmo de treinamento explicado anteriormente.

Em seguida, foi realizado o treinamento da rede usando as funções:

$$\text{net=train(net,Data\_train,Target\_train);}$$

$$\text{Y=sim(net,Data\_train);}$$

A primeira treina a rede neural utilizando os parâmetros passados em *net*. Um desses parâmetros é o número de épocas de treinamento, no qual foram definidos como 500. Enquanto a segunda faz a simulação da rede. Já na Figura 9 é mostrado a performace da rede utilizando os dados de validação e teste.

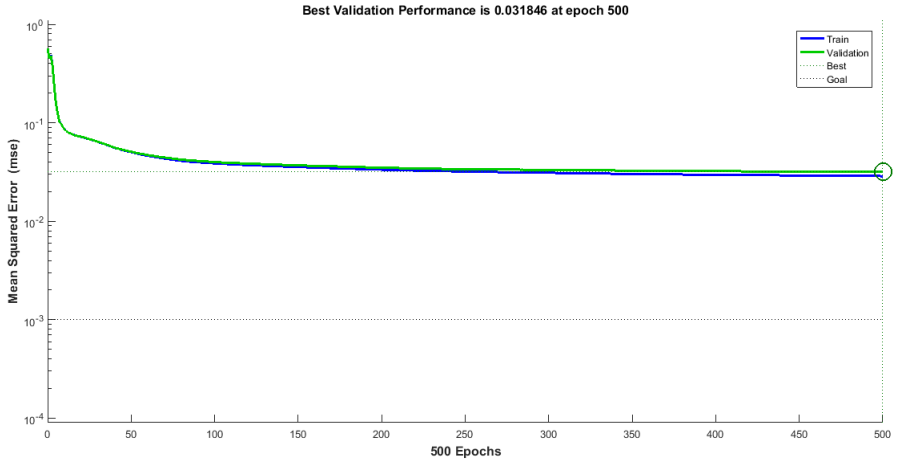

Figura 9: Performace da rede neural.

Após o treinamento é necessário separar os valores obtidos em uma matriz numérica. Assim, é procurado, para cada determinada entrada, a maior nota recebida em relação a todas as saídas do problema e atribui valor 1 a classe com maior valor (código pode ser visualizado em *rede.m*). Depois a mariz de confusão é gerada. Abaixo são apresentadas a matriz de confusão para os dados de treinamento (com 81,9% de acerto) e de teste (com 80,3% de acerto).

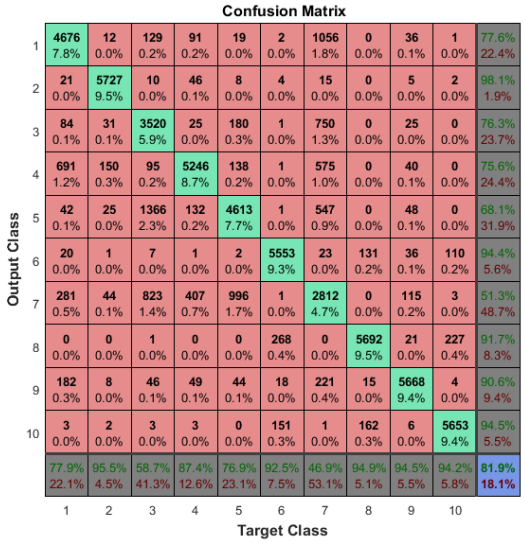

Figura 10: Matriz de confusão com os dados de treino.

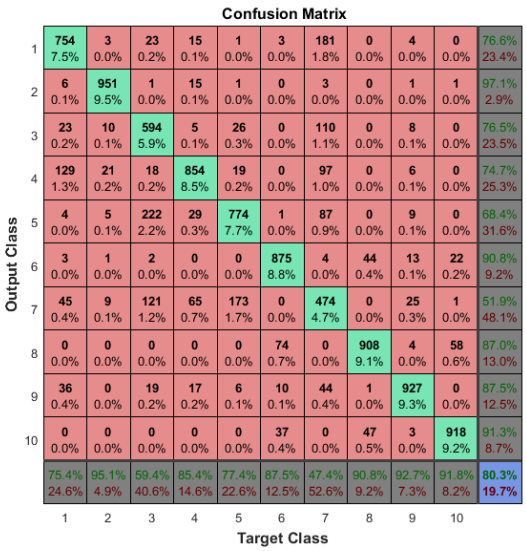

Figura 11: Matriz de confusão com os dados de teste.

Teste 4

Essa rede neural é configurada com a camada de entrada com 784 neurônios, uma camada intermediárias, usando 1 neurônios, e a camada de saída com 10 neurônios.

O algoritmo de treinamento utilizado foi o RProp (trainrp) que é uma função de treinamento de rede que atualiza valores de ponderação e polarização de acordo com o algoritmo resiliente de retropropagação. Foi configurado para tentar alcançar um erro de 0.000001 e uma taxa de aprendizado de 0.05.

$$\text{net} = \text{newff (Data\_train,Target\_train,[1],'tansig' 'purelin','trainrp');}$$

onde os parâmetros são:

- **Data_train:** é o conjunto de dados para treinamento;
- **Target_train:** matriz numérica contendo o dígito que o conjunto de entradas representa;
- **[1]:** número de neurônios nas camada intermediárias;
- **'tansig':** função tangente-simóide de ativação dos neurônios das camadas intermediárias;
- **'purelin':** função de ativação dos neurônios de saída, calcula a saída de uma camada a partir de sua entrada;
- **'trainrp':** algoritmo de treinamento explicado anteriormente.

Em seguida, foi realizado o treinamento da rede usando as funções:

$$\text{net=train(net,Data\_train,Target\_train);}$$

Y=sim(net,Data_train);

A primeira treina a rede neural utilizando os parâmetros passados em *net*. Um desses parâmetros é o número de épocas de treinamento, no qual foram definidos como 500. Enquanto a segunda faz a simulação da rede. A Figura 12 mostra as configurações e o treinamento da rede. Já na Figura 13 é mostrado a performace da rede utilizando os dados de validação e teste.

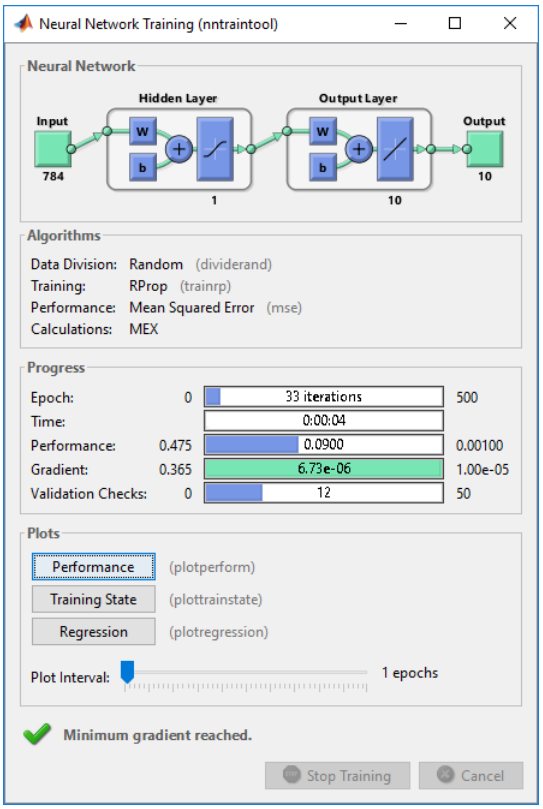

Figura 12: Configurações da rede neural.

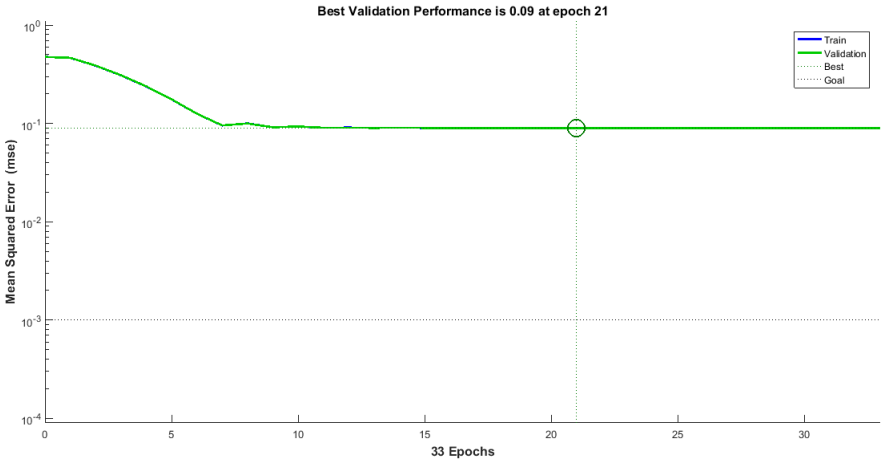

Figura 13: Performace da rede neural.

Após o treinamento é necessário separar os valores obtidos em uma matriz numérica. Assim,

é procurado, para cada determinada entrada, a maior nota recebida em relação a todas as saídas do problema e atribui valor 1 a classe com maior valor (código pode ser visualizado em *rede.m*). Depois a mariz de confusão é gerada. Abaixo são apresentadas a matriz de confusão para os dados de treinamento (com 10% de acerto) e de teste (com 10% de acerto).

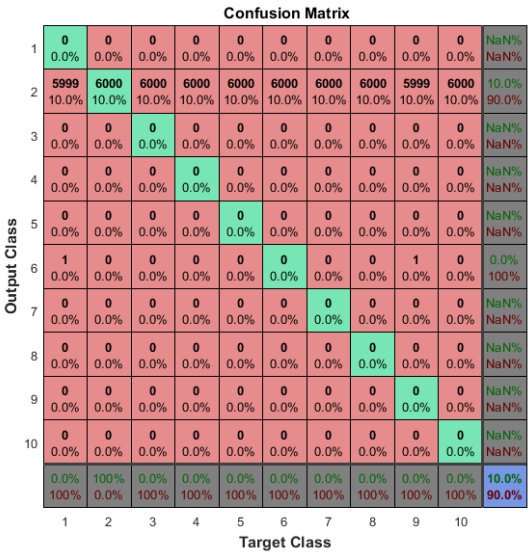

Figura 14: Matriz de confusão com os dados de treino.

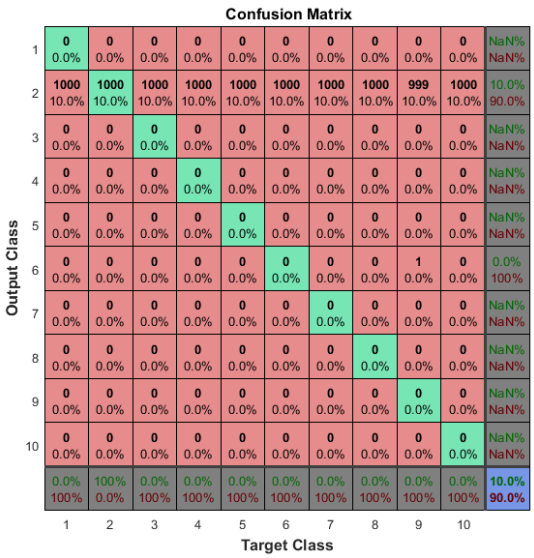

Figura 15: Matriz de confusão com os dados de teste.

Teste 5

Essa rede neural é configurada com a camada de entrada com 784 neurônios, uma camada intermediária, 10 neurônios, e a camada de saída com 10 neurônios.

O algoritmo de treinamento utilizado foi o RProp (trainrp) que é uma função de treinamento de rede que atualiza valores de ponderação e polarização de acordo com o algoritmo resiliente de retropropagação. Foi configurado para tentar alcançar um erro de 0.000001 e uma taxa de aprendizado de 0.05.

$$\text{net} = \text{newff (Data\_train,Target\_train,[10],'tansig' 'purelin','trainrp');}$$

onde os parâmetros são:

- **Data_train:** é o conjunto de dados para treinamento;
- **Target_train:** matriz numérica contendo o dígito que o conjunto de entradas representa;
- **[10]:** número de neurônios nas camada intermediárias;
- **'tansig':** função tangente-simóide de ativação dos neurônios das camadas intermediárias;
- **'purelin':** função de ativação dos neurônios de saída, calcula a saída de uma camada a partir de sua entrada;
- **'trainrp':** algoritmo de treinamento explicado anteriormente.

Em seguida, foi realizado o treinamento da rede usando as funções:

$$\text{net=train(net,Data\_train,Target\_train);}$$

$$\text{Y=sim(net,Data\_train);}$$

A primeira treina a rede neural utilizando os parâmetros passados em *net*. Um desses parâmetros é o número de épocas de treinamento, no qual foram definidos como 500. Enquanto a segunda faz a simulação da rede. A Figura 16 mostra as configurações e o treinamento da rede. Já na Figura 17 é mostrado a performace da rede utilizando os dados de validação e teste.

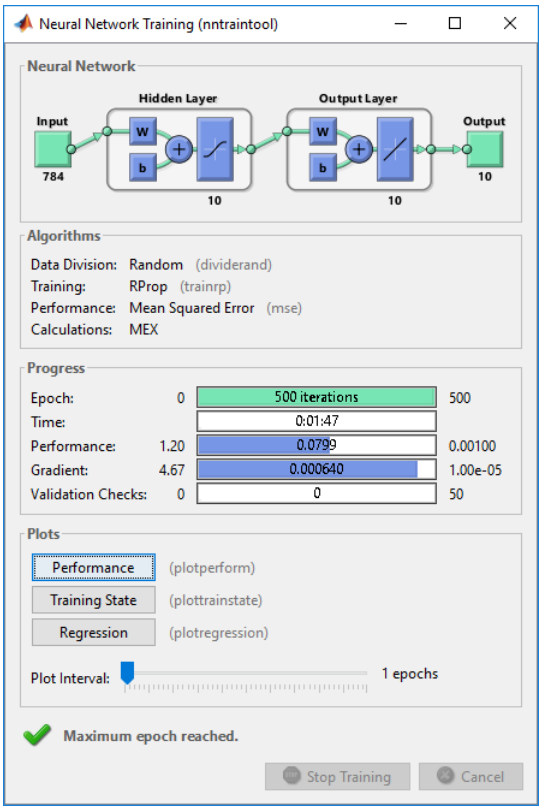

Figura 16: Configurações da rede neural.

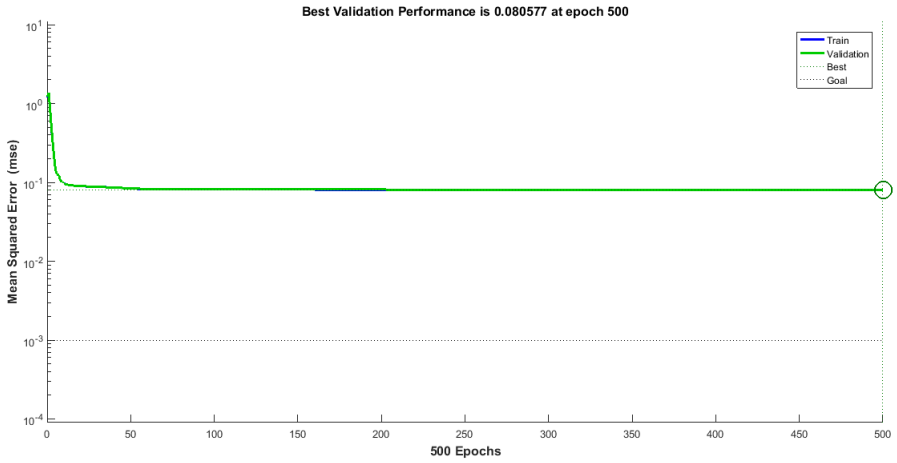

Figura 17: Performace da rede neural.

Após o treinamento é necessário separar os valores obtidos em uma matriz numérica. Assim, é procurado, para cada determinada entrada, a maior nota recebida em relação a todas as saídas do problema e atribui valor 1 a classe com maior valor (código pode ser visualizado em *rede.m*). Depois a mariz de confusão é gerada. Abaixo são apresentadas a matriz de confusão para os dados de treinamento (com 27,5% de acerto) e de teste (com 24,1% de acerto).

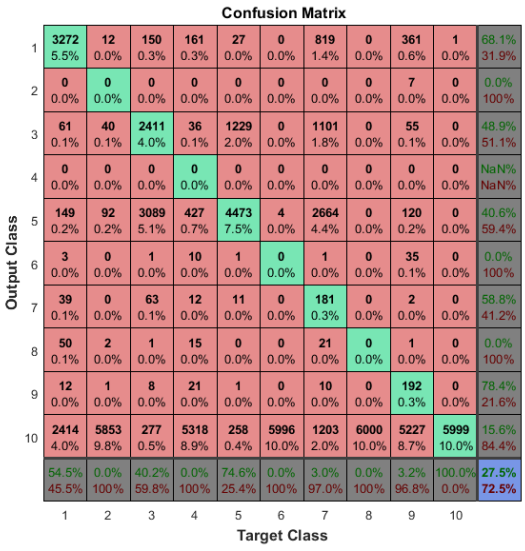

Figura 18: Matriz de confusão com os dados de treino.

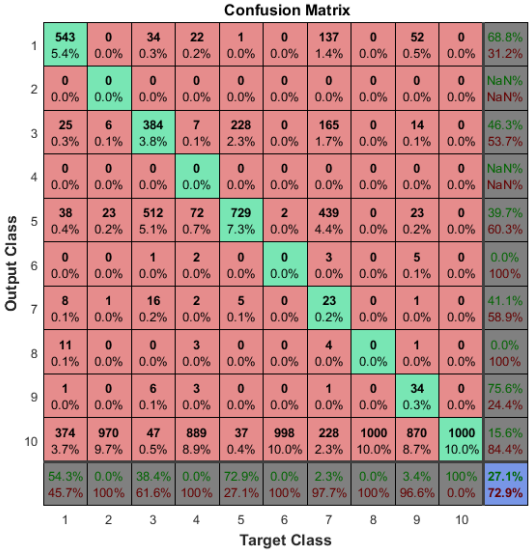

Figura 19: Matriz de confusão com os dados de teste.

3 Conclusão

O desenvolvimento desse trabalho fez com que os membros da equipe aprenderem sobre o treinamento da rede neural. Foram executados vários testes além desses mostrados acima. Foi observado que quando usado redes com apenas uma camada intermediária a taxa de acerto no aprendizado é muito pequena (Testes 4 e 5). As redes neurais que possuem duas camadas de neurônios apresentam uma taxa de acerto boa (Testes 1, 2 e 3). Para melhorar a taxa de acerto poderia ter colocado mais épocas de treinamento, mas como a taxa de acerto usando duas

camadas foi satisfatória então não colocamos mais épocas. Vários erros na detecção de objetos foram encontrados, como no Teste 2, a rede detectou quase todas as peças como 2, assim por falta de neurônios, a rede neural não conseguiu aprender que eram outros tipos de peça.