Rede Neural de Classificação

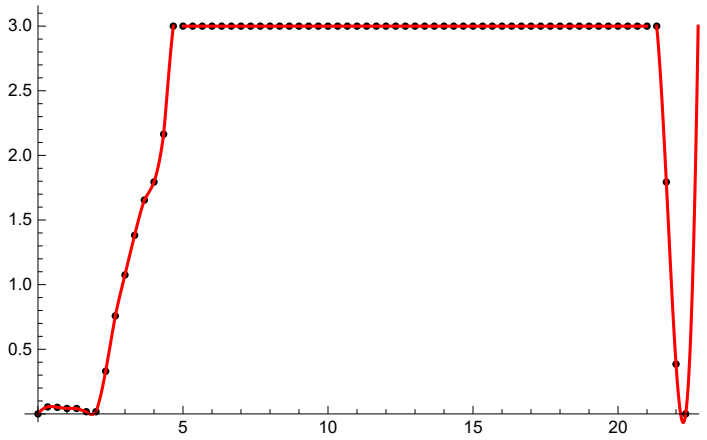
Rotulando os dados

Importando os dados

```
(*--------------Penetração--------------------*)
CP007 = Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\CP007.xlsx"];
CP007 = Flatten[CP007, 1];

CP008 = Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\CP008.xlsx"];
CP008 = Flatten[CP008, 1];

CP009 = Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\CP009.xlsx"];
CP009 = Flatten[CP009, 1];

CP1002 = Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\CP10.02.xlsx"];
CP1002 = Flatten[CP1002, 1];

CP021 = Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\CP021.xlsx"];
CP021 = Flatten[CP021, 1];

CP022 = Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\CP022.xlsx"];
CP022 = Flatten[CP022, 1];

CP045 = Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\CP045.xlsx"];
CP045 = Flatten[CP045, 1];

CP052 = Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\CP052.xlsx"];
CP052 = Flatten[CP052, 1];

(*--------------Botando as penetrações em função do tempo---------------*)

CP007[[All,1]] = CP007[[All,1]] / 2;
CP008[[All,1]] = CP008[[All,1]] / 2;
CP009[[All,1]] = CP009[[All,1]] / 2;
CP1002[[All,1]] = CP1002[[All,1]] / 2;
CP021[[All,1]] = CP021[[All,1]] / 1.7;
CP022[[All,1]] = CP022[[All,1]] / 1.5;
CP045[[All,1]] = CP045[[All,1]] / 1.5;
CP052[[All,1]] = CP052[[All,1]] / 1.5;
```

```
(*--------------- Importando os sons e isolando o processo de soldagem----------------------*

CP007Som = Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\CP007\\MIC1 –
CP007Som = AudioTrim[CP007Som, {1, 26}];

CP008Som = Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\CP008\\MIC1 –
CP008Som = AudioTrim[CP008Som, {2, 17}];

CP009Som = Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\CP009\\MIC1 –
CP009Som = AudioTrim[CP009Som, {1, 23}];

CP1002Som = Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\CP10.02\\MIC
CP1002Som = AudioTrim[CP1002Som, {6, 30}];

CP021Som = Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\CP021\\MIC1 –
CP021Som = AudioTrim[CP021Som, {1, 21}];

CP022Som = Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\CP022\\MIC1 –
CP022Som = AudioTrim[CP022Som, {2, 19}];

CP045Som = Import["C:\\Users\\João\\Documents\\Audacity\\MIC1 – CP45.wav"];
CP045Som = AudioTrim[CP045Som, {3, 21}];

CP052Som = Import["C:\\Users\\João\\Documents\\Audacity\\MIC1 – CP052.wav"];
CP052Som = AudioTrim[CP052Som, {5, 80}];

(*------------------------Penetração em função do tempo------------------*)

PCP007 = CP007;
PCP008 = CP008;
PCP009 = CP009;
PCP1002 = CP1002;
PCP021 = CP021;
PCP022 = CP022;
PCP045 = CP045;
PCP052 = CP052;

(*-------------------------Interpolando----------------------------*)

PCP007Interpolado = Interpolation[PCP007];
PCP008Interpolado = Interpolation[PCP008];
PCP009Interpolado = Interpolation[PCP009];
PCP1002Interpolado = Interpolation[PCP1002];
PCP021Interpolado = Interpolation[PCP021];
PCP022Interpolado = Interpolation[PCP022];
PCP045Interpolado = Interpolation[PCP045];
PCP052Interpolado = Interpolation[PCP052];

(*-------------------------Plotando um grafico-------------------*)
Show[
ListPlot[CP045, PlotStyle→Black],
Plot[PCP045Interpolado[x], {x, 0, 74}, PlotRange→{{0, 70}, {-01, 3}}, PlotStyle→Red]

]
```

Out[ ]=



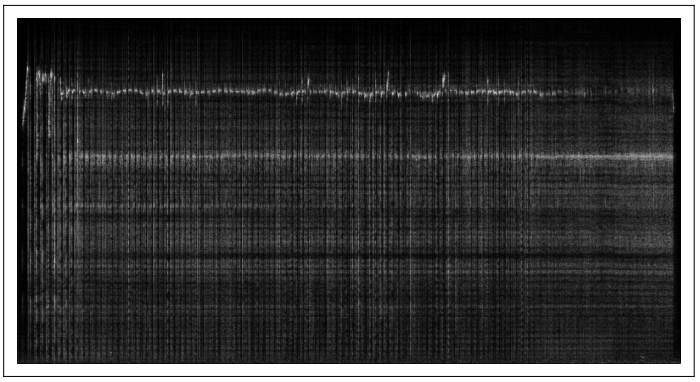
Rotulando os dados em 3 classes

In[ ]:=

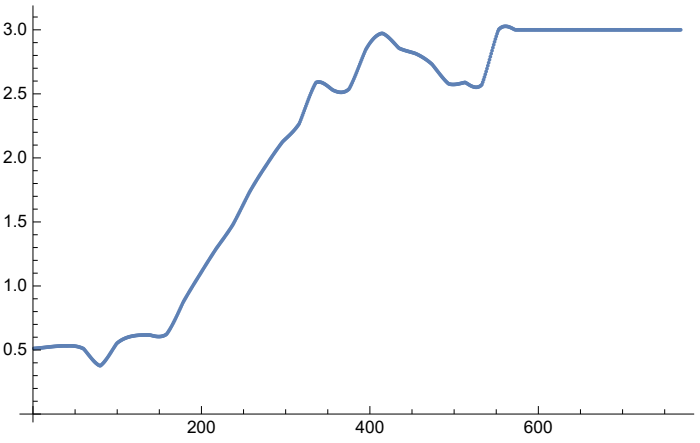
```
(*--------------------------------CP007----------------------------*)

(* Calcula o espectrograma e armazena os dados *)
spec = Spectrogram[CP007Som, 2048, 32, BlackmanHarrisWindow, Frame -> None, ImageSize -> Mediu

(* Extrai a matriz do espectrograma *)
dadosSpec = Transpose[1 - Reverse@spec[[1, 1, All, All, 1]]];
ArrayPlot[dadosSpec // Transpose]
Tempo = Table[i, {i, 0, 25, 25/985}] // N;

PenetrationTempo = Table[
                        {
                            PCP007Interpolado[Tempo[[i]]],

                            If[PCP007Interpolado[Tempo[[i]]]/3*100 ≤ 25, {1,0,0},
                                If[PCP007Interpolado[Tempo[[i]]]/3*100 ≥ 25 && PCP007Interpola
                                If[PCP007Interpolado[Tempo[[i]]]/3*100 ≥ 70, {0, 0, 1}]
                                ]
                            ]
                        },
                    {i, 1, 769}];
(*Criando o dataset*)
dataCP007 = Partition[Transpose[{dadosSpec[[1;;769]], Tempo[[1;;769]], PenetrationTempo}] //
ListPlot[dataCP007[[All, 514]], PlotRange→All]


(*-------------------------------CP052-----------------------------------*)

(* Calcula o espectrograma e armazena os dados *)
spec = Spectrogram[CP052Som, 2048, 32, BlackmanHarrisWindow,
    ColorFunction -> GrayLevel, Frame -> None, ImageSize -> Medium];
```

```
(* Extrai a matriz do espectrograma *)
dadosSpec = Transpose[Reverse@spec[[1, 1, All, All, 1]]];


Tempo = Table[i, {i, 0, 75, 75/994}] // N;
PenetrationTempo = Table[{
                        PCP052Interpolado[Tempo[[i]]],

                        If[PCP052Interpolado[Tempo[[i]]]/3 *100 ≤ 25, {1,0,0},
                              If[PCP052Interpolado[Tempo[[i]]]/3 *100 ≥ 25 && PCP052Interpola
                              If[PCP052Interpolado[Tempo[[i]]]/3 *100 ≥ 70, {0, 0, 1}]
                              ]
                        ]
                    },
                  {i, 1, 981}];


dataCP052 = Partition[Transpose[{dadosSpec[[1;;981]], Tempo[[1;;981]], PenetrationTempo}] //
```

Out[ ]=

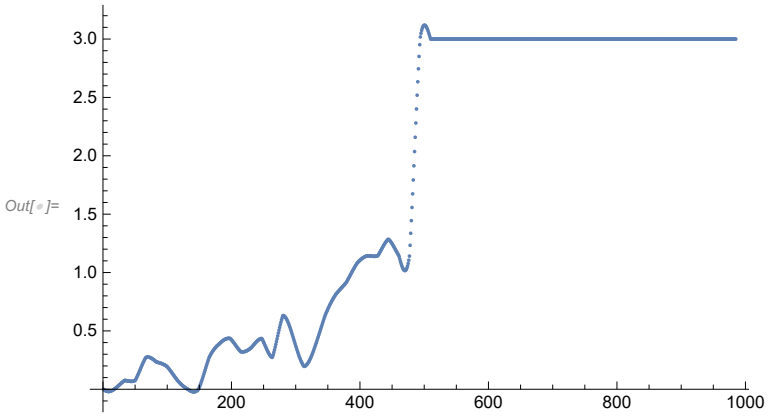


Out[ ]=



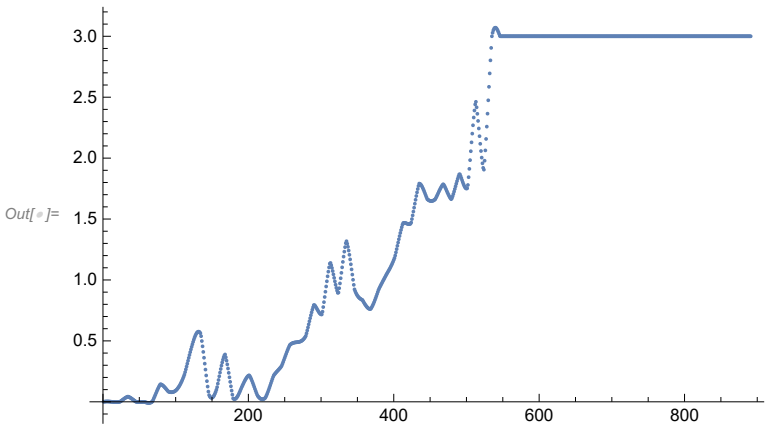
In[ ]:=
```
(*--------------------------------CP008---------------------------*)

(* Calcula o espectrograma e armazena os dados *)
spec = Spectrogram[CP008Som, 2048, 32, BlackmanHarrisWindow,
    ColorFunction -> GrayLevel, Frame -> None, ImageSize -> Medium];

(* Extrai a matriz do espectrograma *)
dadosSpec = Transpose[1 - Reverse@spec[[1, 1, All, All, 1]]];
Tempo = Table[i, {i, 0, 15, 15/985}] // N;

PenetrationTempo = Table[
                        {
                            PCP008Interpolado[Tempo[[i]]],

                            If[PCP008Interpolado[Tempo[[i]]]/3*100 ≤ 25, {1,0,0},
                                If[PCP008Interpolado[Tempo[[i]]]/3*100 ≥ 25 && PCP008Interpola
                                If[PCP008Interpolado[Tempo[[i]]]/3*100 ≥ 70, {0, 0, 1}]
                                ]
                            ]
                        },
                    {i, 1, 985}];
(*Criando o dataset*)
dataCP008 = Partition[Transpose[{dadosSpec[[1;;985]], Tempo[[1;;985]], PenetrationTempo}] //
ListPlot[dataCP008[[All, 514]], PlotRange→All]
```

Out[ ]=



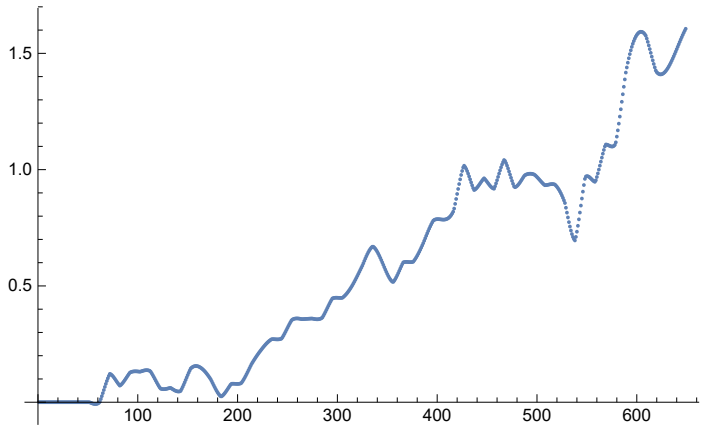
In[ ]:=
```
(*--------------------------------CP009---------------------------*)

(* Calcula o espectrograma e armazena os dados *)
spec = Spectrogram[CP009Som, 2048, 32, BlackmanHarrisWindow,
    ColorFunction -> GrayLevel, Frame -> None, ImageSize -> Medium];

(* Extrai a matriz do espectrograma *)
dadosSpec = Transpose[1 - Reverse@spec[[1, 1, All, All, 1]]];
Tempo = Table[i, {i, 0, 22, 22/979}] // N;

PenetrationTempo = Table[
                        {
                            PCP009Interpolado[Tempo[[i]]],

                            If[PCP009Interpolado[Tempo[[i]]]/3*100 ≤ 25, {1,0,0},
                                If[PCP009Interpolado[Tempo[[i]]]/3*100 ≥ 25 && PCP009Interpola
                                If[PCP009Interpolado[Tempo[[i]]]/3*100 ≥ 70, {0, 0, 1}]
                                ]
                            ]
                        },
                    {i, 1, 891}];
(*Criando o dataset*)
dataCP009 = Partition[Transpose[{dadosSpec[[1;;891]], Tempo[[1;;891]], PenetrationTempo}] //
ListPlot[dataCP009[[All, 514]], PlotRange→All]
```

Out[ ]=



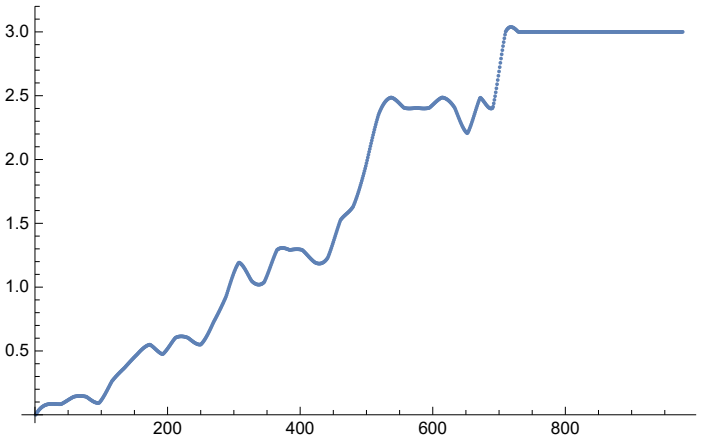
In[ ]:=
```
(*--------------------------------CP10.02---------------------------*)

(* Calcula o espectrograma e armazena os dados *)
spec = Spectrogram[CP1002Som, 2048, 32, BlackmanHarrisWindow,
    ColorFunction -> GrayLevel, Frame -> None, ImageSize -> Medium];

(* Extrai a matriz do espectrograma *)
dadosSpec = Transpose[1 - Reverse@spec[[1, 1, All, All, 1]]];
Tempo = Table[i, {i, 0, 24, 24/973}] // N;

PenetrationTempo = Table[
                        {
                            PCP1002Interpolado[Tempo[[i]]],

                            If[PCP1002Interpolado[Tempo[[i]]]/3*100 ≤ 25, {1,0,0},
                                If[PCP1002Interpolado[Tempo[[i]]]/3*100 ≥ 25 && PCP1002Interpo
                                If[PCP1002Interpolado[Tempo[[i]]]/3*100 ≥ 70, {0, 0, 1}]
                                ]
                            ]
                        },
                    {i, 1, 649}];
(*Criando o dataset*)
dataCP1002 = Partition[Transpose[{dadosSpec[[1;;649]], Tempo[[1;;649]], PenetrationTempo}] //
ListPlot[dataCP1002[[All, 514]], PlotRange→All]
```

Out[ ]=



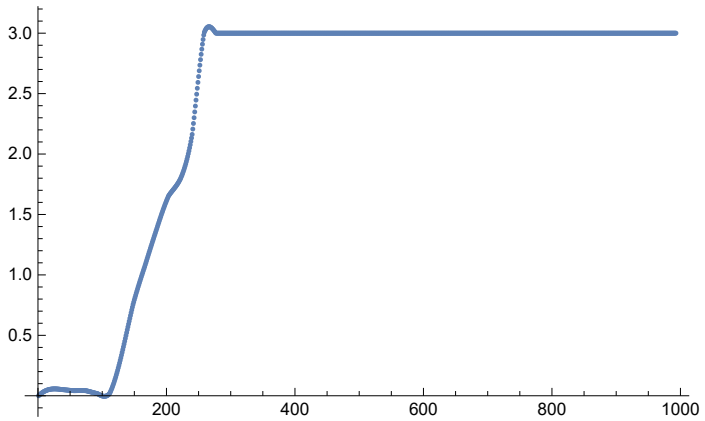
In[ ]:=
```
(*--------------------------------CP022----------------------------*)

(* Calcula o espectrograma e armazena os dados *)
spec = Spectrogram[CP022Som, 2048, 32, BlackmanHarrisWindow,
    ColorFunction -> GrayLevel, Frame -> None, ImageSize -> Medium];

(* Extrai a matriz do espectrograma *)
dadosSpec = Transpose[1 - Reverse@spec[[1, 1, All, All, 1]]];
Tempo = Table[i, {i, 0, 17, 17/Length[dadosSpec]}] // N;

PenetrationTempo = Table[
                        {
                            PCP022Interpolado[Tempo[[i]]],

                            If[PCP022Interpolado[Tempo[[i]]]/3*100 ≤ 25, {1,0,0},
                                If[PCP022Interpolado[Tempo[[i]]]/3*100 ≥ 25 && PCP022Interpola
                                If[PCP022Interpolado[Tempo[[i]]]/3*100 ≥ 70, {0, 0, 1}]
                                ]
                            ]
                        },
                    {i, 1, 977}];
(*Criando o dataset*)
dataCP022 = Partition[Transpose[{dadosSpec[[1;;977]], Tempo[[1;;977]], PenetrationTempo}] //
ListPlot[dataCP022[[All, 514]], PlotRange→All]
```

Out[ ]=



In[ ]:=
```
(*--------------------------------CP045---------------------------*)

(* Calcula o espectrograma e armazena os dados *)
spec = Spectrogram[CP045Som, 2048, 32, BlackmanHarrisWindow,
    ColorFunction -> GrayLevel, Frame -> None, ImageSize -> Medium];

(* Extrai a matriz do espectrograma *)
dadosSpec = Transpose[1 - Reverse@spec[[1, 1, All, All, 1]]];
Tempo = Table[i, {i, 0, 18, 18/Length[dadosSpec]}] // N;

PenetrationTempo = Table[
                        {
                            PCP045Interpolado[Tempo[[i]]],

                            If[PCP045Interpolado[Tempo[[i]]]/3*100 ≤ 25, {1,0,0},
                                If[PCP045Interpolado[Tempo[[i]]]/3*100 ≥ 25 && PCP045Interpola
                                If[PCP045Interpolado[Tempo[[i]]]/3*100 ≥ 70, {0, 0, 1}]
                                ]
                            ]
                        },
                    {i, 1, 993}];
(*Criando o dataset*)
dataCP045 = Partition[Transpose[{dadosSpec[[1;;993]], Tempo[[1;;993]], PenetrationTempo}] //
ListPlot[dataCP045[[All, 514]], PlotRange→All]
```

Out[ ]=



Criando o DataSet

In[ ]:=
```
dataset = Join[dataCP007, dataCP008, dataCP009, dataCP1002, dataCP022, dataCP045, dataCP052];
```

Rede neural de classificação

Rede neural de classificação

In[ ]:=

```
GerarPesos[linhas_, colunas_, gerarpesos_, camada_] :=
Module[{i = linhas, j = colunas, condicao = ToString[gerarpesos],
        indice = camada},
(*Pesos*)
For[linha = 1, linha <= (i-1), linha++,

    For[coluna = 1, coluna <= j, coluna++,
        If[condicao == "sim",
        w[indice][linha,coluna] = RandomReal[{-0.5, 0.5}], Clear[w]
        ]
    ]
];
(*Bias*)
For[linha = i, linha <= i, linha++,
    For[coluna = 1, coluna <= j, coluna++,
        If[condicao == "sim", w[indice][linha, coluna] = 1,
        ClearAll
        ]
    ]
];

Table[w[indice][m, n], {m, 1, i}, {n, 1, j}]
]

(*Funcao de ativacao na camada de output*)
Softmax[x_] := e^x / (∑_{i=1}^{Length[x]} e^{x[[i]]})
(*Funcao de ativacao nas camadas ocultas*)
R[x_] := Max[0.01*x, x]

NeuroniosCamada1 = 10;
NeuroniosCamada2 = 8;
NeuroniosCamada3 = 7;
NeuroniosCamada4 = 5;

(*Pesos da primeira camada oculta*)
    (*A linha 22050 é a entrada para as amplitudes, e a última,
    22051 no caso, é o bias*)
    W1[0] = GerarPesos[513, NeuroniosCamada1, sim, 1];

(*Pesos da segunda camada oculta*)
    (*A camada seguinte tem q levar em consideração
    o bias da camada anterior, por isso adciona-se 1*)
    W2[0] = GerarPesos[NeuroniosCamada1+1, NeuroniosCamada2, sim, 2];

(*Pesos da terceira camada oculta*)
    W3[0] = GerarPesos[NeuroniosCamada2+1, NeuroniosCamada3, sim, 2];

(*Pesos da quarta camada oculta*)
```

```
    W4[0] = GerarPesos[NeuroniosCamada3+1, NeuroniosCamada4, sim, 2];

(*Pesos da quinta camada oculta*)
    W5[0] = GerarPesos[NeuroniosCamada4+1, 3, sim, 2];
```

Criando mini-batchs

```
datasetRandom = RandomSample[dataset];

(*Datatrain contento 70% do dataset*)
datatrain = Take[datasetRandom, Round[70/100*Length[dataset]]];
(*Datateste contendo os 30% restantes do dataset*)
datatest = Drop[datasetRandom, Round[70/100*Length[dataset]]];
(*MiniBatch*)
data = Partition[datatrain, 2^5, 2^5, 1, {}];
Table[minibatch[i] = Flatten[Take[data, {i, i}], 1], {i, 1, Length[data]}];
```

Treinando a Rede Neural de classificação

In[•]:=

```
epocas = 10000;
α = 0.0001;
θ = 0.5;
numbatch = 1;
(*Dynamic[Text["Cross Entropy Treino: " <> ToString[CrossEntropy[k]] <> " | Cross Entropy Tes
" | Epoca: " <> ToString[k]] ]*)

For[k = 0, k ≤ epocas, k++,

    If[numbatch == Length[data]+1, numbatch = 1];
    batchsize = Length[minibatch[numbatch]];

    If[Mod[k, 1000] == 0,
    CrossEntropy[k] = ((-1)/Length[datatrain])*Sum[datatrain[[p, 515;;517]].Log[Softmax[Fl, {p, 1, Length[datatrain]}]
    crossEntropytest[k] = ((-1)/Length[datatest])*Sum[datatest[[p, 515;;517]].Log[Softmax[F, {p, 1, Length[datatest]}]
    Print["Cross Entropy Treino: ", CrossEntropy[k], " | Cross Entropy Teste: ", crossEntropyt
    ];

    For[j = 1, j ≤ batchsize, j++,

        x = Partition[Join[minibatch[numbatch][[j, 1;;512]], {1}], 1];
        z[1] = Transpose[W1[k]].x;
        a[1] = Partition[Join[Map[R, z[1]], {1}], 1];

        z[2] = Transpose[W2[k]].a[1];
        a[2] = Partition[Join[Map[R, z[2]], {1}], 1];

        z[3] = Transpose[W3[k]].a[2];
        a[3] = Partition[Join[Map[R, z[3]], {1}], 1];
```

```
    z[4] = Transpose[W4[k]].a[3];
    a[4] = Partition[Join[Map[R, z[4]], {1}], 1];

    z[5] = Transpose[W5[k]].a[4];
    a[5] = Partition[Softmax[Flatten[z[5]]], 1];

(*BackPropagation*)

    If[minibatch[numbatch][[j, 515;;517]] == {1, 0, 0}, δ[5] = a[5] - {1, 0, 0}];
    If[minibatch[numbatch][[j, 515;;517]] == {0, 1, 0}, δ[5] = a[5] - {0, 1, 0}];
    If[minibatch[numbatch][[j, 515;;517]] == {0, 0, 1}, δ[5] = a[5] - {0, 0, 1}];

        (*Camada 5*)
    (*Atualizando W5*)
    ∇W5 = a[4].Transpose[δ[5]];
    If[k ≥ 2 && j ≥ 2 && numbatch-1 ≥ 2, v5 = θ*(W5[k,j] - W5[k-1,j]), v5 = 0];
    W5[k+1,j] = W5[k] + v5 - α*∇W5;

        (*Camada 4*)
    δ[4] = (W5[k][[1;;NeuroniosCamada4]].δ[5])*Map[R', Flatten[z[4]]];
    (*Atualizando W4*)
    ∇W4 = a[3].Transpose[δ[4]];
    If[k ≥ 2 && j ≥ 2 && numbatch-1 ≥ 2, v4 = θ*(W4[k,j] - W4[k-1,j]), v4 = 0];
    W4[k+1,j] = W4[k] + v4 - α*∇W4;

        (*Camada 3*)
    δ[3] = (W4[k][[1;;NeuroniosCamada3]].δ[4])*Map[R', Flatten[z[3]]];
    (*Atualizando W3*)
    ∇W3 = a[2].Transpose[δ[3]];
    If[k ≥ 2 && j ≥ 2  && numbatch-1 ≥ 2, v3 = θ*(W3[k,j] - W3[k-1,j]), v3 = 0];
    W3[k+1,j]= W3[k] + v3 - α*∇W3;

        (*Camada 2*)
    δ[2] = (W3[k][[1;;NeuroniosCamada2]].δ[3])*Map[R', Flatten[z[2]]];
    (*Atualizando W2*)
    ∇W2 = a[1].Transpose[δ[2]];
    If[k ≥ 2 && j ≥ 2 && numbatch-1 ≥ 2, v2 = θ*(W2[k,j] - W2[k-1,j]), v2 = 0];
    W2[k+1,j] = W2[k] + v2 - α*∇W2;

        (*Camada 1*)
    δ[1] = (W2[k][[1;;NeuroniosCamada1]].δ[2])*Map[R', Flatten[z[1]]];
    (*Atualizando W1*)
    ∇W1 = x.Transpose[δ[1]];
    If[k ≥ 2 && j ≥ 2 && numbatch-1 ≥ 2, v1 = θ*(W1[k,j] - W1[k-1,j]), v1 = 0];
    W1[k+1,j] = W1[k] + v1 - α*∇W1;

];
W1[k+1] = Mean[Table[W1[k+1, m], {m, 1, batchsize}]];
W2[k+1] = Mean[Table[W2[k+1, m], {m, 1, batchsize}]];
W3[k+1] = Mean[Table[W3[k+1, m], {m, 1, batchsize}]];
W4[k+1] = Mean[Table[W4[k+1, m], {m, 1, batchsize}]];
W5[k+1] = Mean[Table[W5[k+1, m], {m, 1, batchsize}]];
```

```
    numbatch = numbatch + 1;
]
```

Salvando o modelo

In[ ]:=
```
Export["W4 - Classificacao.xlsx", W4[0]];
Export["W3 - Classificacao.xlsx", W3[0]];
Export["W2 - Classificacao.xlsx", W2[0]];
Export["W1 - Classificacao.xlsx", W1[0]];
```

Matriz de confusão

```
k = 0;

RespostasRN = Table[Softmax[Flatten[Transpose[W5[k]].Partition[Join[Map[R, Transpose[W4[k]].P
RespostasRN = Table[Position[RespostasRN[[i]], Max[RespostasRN[[i]]] ][[1,1]], {i, 1, datates

respostas = Table[datatest[[p, 515;;517]], {p, 1, datatest // Length}] /. {{1, 0, 0} → "Confo
```

In[ ]:=
```
tabela = Transpose[{RespostasRN, respostas}];
tabelacomparacao = TableForm[tabela, TableHeadings→{None, {"Resposta Rede Neural", "Resposta C
```

In[ ]:=

```
FuroAcertos = 0;
QuaseFuroAcertos = 0;
ConformeAcertos = 0;

Furo_QuaseFuro = 0;
Furo_Conforme = 0;

QuaseFuro_Furo = 0;
QuaseFuro_Conforme = 0;

Conforme_Furo = 0;
Conforme_QuaseFuro = 0;

For[i = 1, i ≤ Length[datatest], i++,
    If[respostas[[i]] == RespostasRN[[i]] == "Furo", FuroAcertos = FuroAcertos + 1];
    If[respostas[[i]] == RespostasRN[[i]] == "Quase Furo", QuaseFuroAcertos = QuaseFuroAcertos
    If[respostas[[i]] == RespostasRN[[i]] == "Conforme", ConformeAcertos = ConformeAcertos + 1

    If[respostas[[i]] == "Furo" && RespostasRN[[i]] == "Quase Furo", Furo_QuaseFuro = Furo_QuaseFuro
    If[respostas[[i]] == "Furo" && RespostasRN[[i]] == "Conforme", Furo_Conforme = Furo_Conforme + 1

    If[respostas[[i]] == "Quase Furo" && RespostasRN[[i]] == "Furo", QuaseFuro_Furo = QuaseFuro_F
    If[respostas[[i]] == "Quase Furo" && RespostasRN[[i]] == "Conforme", QuaseFuro_Conforme = Qua

    If[respostas[[i]] == "Conforme" && RespostasRN[[i]] == "Furo", Conforme_Furo = Conforme_Furo +
    If[respostas[[i]] == "Conforme" && RespostasRN[[i]] == "Quase Furo", Conforme_QuaseFuro = Con
]

Acuracia = (FuroAcertos + QuaseFuroAcertos + ConformeAcertos)/Length[datatest]*100 // N;
```

In[•]:=
```
matrizConfusao = Grid[{
                        {"","Furo", "Quase Furo", "Conforme"},
                        {"Furo Previsão", FuroAcertos, QuaseFuro_Furo, Conforme_Furo},
                        {"Quase Furo Previsão", Furo_QuaseFuro, QuaseFuroAcertos, Conforme_QuaseFuro},
                        {"Conforme Previsão", Furo_Conforme, QuaseFuro_Conforme, ConformeAcertos},
                        {"Acurácia", Acuracia "%"},
                        {, "Metricas de avaliação"},
                        {,"Furo", "Quase Furo", "Conforme"},
                        {"Sensibilidade", FuroAcertos / (FuroAcertos + Furo_QuaseFuro + Furo_Conforme) // N[#, 3]
                        {"Especificidade", (QuaseFuroAcertos+ConformeAcertos) / (QuaseFuroAcertos+ConformeAcertos+QuaseFuro_Furo+Conf
                        {"Valor Preditivo Positivo", FuroAcertos / (FuroAcertos+QuaseFuro_Furo+Conforme_Furo)//N|
                        {"Valor Preditivo Negativo", (QuaseFuroAcertos+ConformeAcer / (QuaseFuroAcertos+ConformeAcertos+Furo_Quas
                        {"Precision", FuroAcertos / (FuroAcertos + QuaseFuro_Furo + Conforme_Furo)//N[#, 3] &, —
                                                                                                                Q
                        {"F1-Score", 2*(FuroAcertos / (FuroAcertos + QuaseFuro_Furo + Conforme_Furo))*(FuroAcertos / (FuroAcertos + Furo_QuaseFuro + F
                                     (FuroAcertos / (FuroAcertos + QuaseFuro_Furo + Conforme_Furo)) + (FuroAcertos / (FuroAcertos + Furo_QuaseFuro + Fu
                      },

                          Frame → True,
                          Dividers →{True, {6 → True, 7 → True}},
                          Spacings → {2, 1}(*,
                          Dividers → Center*)
                      ]
(*Salvando essa matriz*)
Export["Matriz de confusão.PNG", matrizConfusao]
```

	Furo	Quase Furo	Conforme
Furo Previsão	596	58	40
Quase Furo Previsão	25	55	15
Conforme Previsão	35	25	1024
Acurácia	89.4287 %		
Metricas de avaliação			
	Furo	Quase Furo	Conforme
Sensibilidade	0.909	0.399	0.949
Especificidade	0.917	0.976	0.916
Valor Preditivo Positivo	0.859	0.579	0.945
Valor Preditivo Negativo	0.947	0.951	0.922
Precision	0.859	0.579	0.945
F1-Score	0.883	0.472	0.947

Out[•]= Matriz de confusão.PNG

Gerando a curva de perda

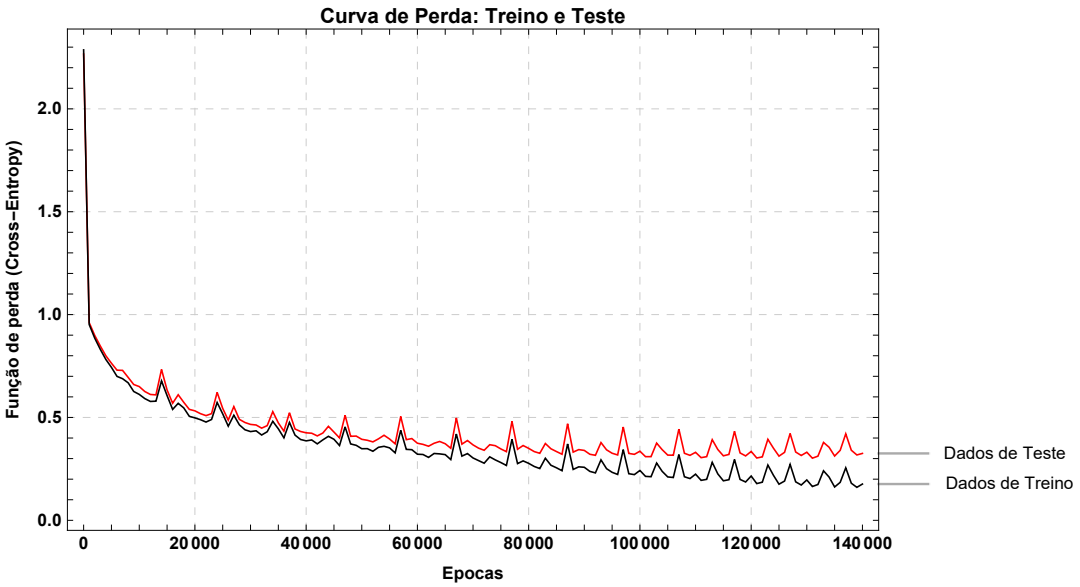
```
Loss = Flatten[Import["C:\\Users\\João\\Desktop\\Wolfram projetos\\Soldagem Wolfram\\Loss ao l
```

In[ ]:=
```
iteracoes = Table[i, {i, 0, 141*1000, 1000}];

CrossEntropydatatrain = ToExpression[Transpose[{iteracoes[[1;;141]], Loss[[All,1]][[1;;141]]}
CrossEntropydatatest = ToExpression[Transpose[{iteracoes[[1;;141]], Loss[[All,2]][[1;;141]]}]
```

Plotando o gráfico

In[ ]:=
```
p = ListPlot[
            Labeled[CrossEntropydatatest, "Dados de Teste"],
            ImageSize→Large,
            Frame→True,
            FrameLabel → {"Epocas", "Função de perda (Cross-Entropy)"},
            FrameStyle →Directive[Black,Bold],
            PlotStyle→{Red, Thickness[0.002]},
            Joined→True,
            GridLines→Automatic,
            GridLinesStyle -> Directive[GrayLevel[0.8], Dashed, Thin],
            PlotLabel → Style["Curva de Perda: Treino e Teste", Bold],
            PlotRange→Full
            ];

q = ListPlot[
            Labeled[CrossEntropydatatrain, "Dados de Treino"],
            ImageSize→Large,
            Frame→True,
            FrameLabel → {"Epocas", "Função de perda (Cross-Entropy)"},
            FrameStyle→Directive[Black,Bold],
            PlotStyle → {Black, Thickness[0.002]},
            Joined→True,
            GridLines→Automatic,
            GridLinesStyle -> Directive[GrayLevel[0.8], Dashed, Thin],
            PlotLabel → Style["Curva de Perda: Treino e Teste", Bold],
            PlotRange→Full
            ];

grafico = Show[p, q]
```

Out[ ]=



Testando o modelo no CP007

In[ ]:=

```
(*--------------------------------CP007----------------------------*)

(* Calcula o espectrograma e armazena os dados *)
spec = Spectrogram[CP007Som, 2048, 32, BlackmanHarrisWindow,
    ColorFunction -> GrayLevel, Frame -> None, ImageSize -> Medium];

(* Extrai a matriz do espectrograma *)
dadosSpec = Transpose[1 - Reverse@spec[[1, 1, All, All, 1]]];
ArrayPlot[dadosSpec // Transpose]
Tempo = Table[i, {i, 0, 25, 25/985}] // N;

PenetrationTempo = Table[
                        {
                            PCP007Interpolado[Tempo[[i]]],

                            If[PCP007Interpolado[Tempo[[i]]]/3*100 ≤ 25, {1,0,0},
                                If[PCP007Interpolado[Tempo[[i]]]/3*100 ≥ 25 && PCP007Interpola
                                If[PCP007Interpolado[Tempo[[i]]]/3*100 ≥ 70, {0, 0, 1}]
                                ]
                            ]
                        },
                    {i, 1, 769}];
(*Criando o dataset*)
dataCP007 = Partition[Transpose[{dadosSpec[[1;;769]], Tempo[[1;;769]], PenetrationTempo}] //
ListPlot[dataCP007[[All, 514]], PlotRange→All]
```

```
RespostasRN = Table[Softmax[Flatten[Transpose[W5[k]].Partition[Join[Map[R, Transpose[W4[k]].P
RespostasRN = Table[Position[RespostasRN[[i]], Max[RespostasRN[[i]]] ][[1,1]], {i, 1, 769}]

respostas = Table[dataCP007[[p, 515;;517]], {p, 1, dataCP007 // Length}] /. {{1, 0, 0} → "Cor


tabela = Transpose[{RespostasRN, respostas}];
tabelacomparacao = TableForm[tabela, TableHeadings→{None, {"Resposta Rede Neural", "Resposta C
```

In[ ]:=

```
FuroAcertos = 0;
QuaseFuroAcertos = 0;
ConformeAcertos = 0;

Furo_QuaseFuro = 0;
Furo_Conforme = 0;

QuaseFuro_Furo = 0;
QuaseFuro_Conforme = 0;

Conforme_Furo = 0;
Conforme_QuaseFuro = 0;

For[i = 1, i ≤ Length[dataCP007], i++,
    If[respostas[[i]] == RespostasRN[[i]] == "Furo", FuroAcertos = FuroAcertos + 1];
    If[respostas[[i]] == RespostasRN[[i]] == "Quase Furo", QuaseFuroAcertos = QuaseFuroAcertos
    If[respostas[[i]] == RespostasRN[[i]] == "Conforme", ConformeAcertos = ConformeAcertos + 1

    If[respostas[[i]] == "Furo" && RespostasRN[[i]] == "Quase Furo", Furo_QuaseFuro = Furo_QuaseFuro
    If[respostas[[i]] == "Furo" && RespostasRN[[i]] == "Conforme", Furo_Conforme = Furo_Conforme + 1

    If[respostas[[i]] == "Quase Furo" && RespostasRN[[i]] == "Furo", QuaseFuro_Furo = QuaseFuro_F
    If[respostas[[i]] == "Quase Furo" && RespostasRN[[i]] == "Conforme", QuaseFuro_Conforme = Qua

    If[respostas[[i]] == "Conforme" && RespostasRN[[i]] == "Furo", Conforme_Furo = Conforme_Furo +
    If[respostas[[i]] == "Conforme" && RespostasRN[[i]] == "Quase Furo", Conforme_QuaseFuro = Con
]

Acuracia = (FuroAcertos + QuaseFuroAcertos + ConformeAcertos)/Length[dataCP007]*100 // N;
```

In[ ]:=

```
matrizConfusao = Grid[{
                        {"","Furo", "Quase Furo", "Conforme"},
                        {"Furo Previsão", FuroAcertos, QuaseFuro_Furo, Conforme_Furo},
                        {"Quase Furo Previsão", Furo_QuaseFuro, QuaseFuroAcertos, Conforme_QuaseFuro
                        {"Conforme Previsão", Furo_Conforme, QuaseFuro_Conforme, ConformeAcertos},
                        {"Acurácia", Acuracia "%", , Style["CP007", Red, Bold]},
                        {, "Metricas de avaliação"},
                        {,"Furo", "Quase Furo", "Conforme"},
                        {"Sensibilidade", FuroAcertos / (FuroAcertos + Furo_QuaseFuro + Furo_Conforme // N[#, 3]
                        {"Especificidade", (QuaseFuroAcertos+ConformeAcertos) / (QuaseFuroAcertos+ConformeAcertos+QuaseFuro_Furo+Conf
                        {"Valor Preditivo Positivo", FuroAcertos / (FuroAcertos+QuaseFuro_Furo+Conforme_Furo //N|
                        {"Valor Preditivo Negativo", (QuaseFuroAcertos+ConformeAcer) / (QuaseFuroAcertos+ConformeAcertos+Furo_Quas
                        {"Precision", FuroAcertos / (FuroAcertos + QuaseFuro_Furo + Conforme_Furo //N[#, 3] &, -
                                                                                                            Q
                        {"F1-Score", 2*(FuroAcertos / (FuroAcertos + QuaseFuro_Furo + Conforme_Furo))*(FuroAcertos / (FuroAcertos + Furo_QuaseFuro + F
                                     (FuroAcertos / (FuroAcertos + QuaseFuro_Furo + Conforme_Furo)) + (FuroAcertos / (FuroAcertos + Furo_QuaseFuro + Fu
                      },

                          Frame → True,
                          Dividers →{True, {6 → True, 7 → True}},
                          Spacings → {2, 1}(*,
                          Dividers → Center*)
                        ]

(*Salvando essa matriz*)
Export["Matriz de confusão - CP007.PNG", matrizConfusao]
```