Não definido

Rodrigo C. Moraes, Maurício C. Figueiredo, Elloá B. Guedes

{rcm, cfigueiredo, ebgcosta}.eng@uea.edu.br

[1]Núcleo de Computação
Escola Superior de Tecnologia
Universidade do Estado do Amazonas
Av. Darcy Vargas, 1200 – Manaus – Amazonas

Abstract. *Text*

Resumo. *Texto*

1. Introdução

Texto

2. Materiais e Métodos

2.1. Base de dados

Como base de dados utilizou-se a *Facial Expression Recognition Challenge*, conhecida como *FER2013*. Que é disponibilizada pela plataforma de competição em *Machine Learning*, *Kaggle*. E consiste de imagens faciais em escala de cinza, onde cada expressão contida na imagem é classificada como uma das Sete Expressões Faciais Universais []. No total, são 35887 imagens de exemplo, dividas de por expressão: 4953 para raiva, 547 para nojo, 5121 para medo, 8989 para felicidade, 6077 para tristeza, 4002 para surpresa e 6198 para neutro. A distribuição das quantidade de exemplos por expressão podem ser melhor visualizadas na Figura 1, onde quanto mais escura a cor da barra, maior a quantidade de exemplos de determinada expressão.

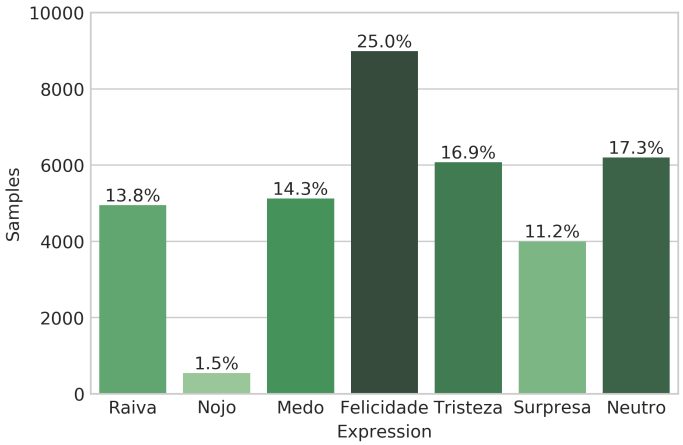

Figure 1. Distribuição de imagens por Expressão

Além das divisões das expressões, a base de dados é divida em três partes: *Training*, *PublicTest* e *PrivateTest*. Está foi realizada pelos pesquisadores [], responsáveis pela competição que incluía essa base. Cada parte possui finalidade específica, *Training* deve ser usada para treinamento, *PublicTest* para validação e *PrivateTest* para teste dos modelos. Durante a competição os exemplos com *label PrivateTest* não possuíam classificação, contudo, estes foram disponibilizados posteriormente para que fosse possível avaliar o desempenho de modelos externos a competição. A distribuição dos dados pode ser visualizada nas Figuras 2, 3 e 4.

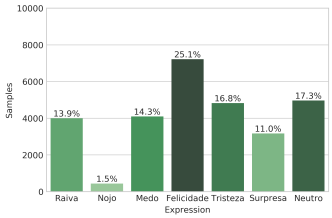

Figure 2. Distribuição *Training*

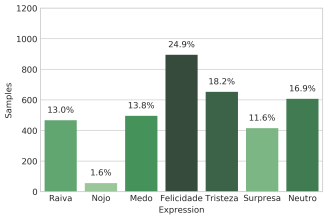

Figure 3. Distribuição *PublicTest*

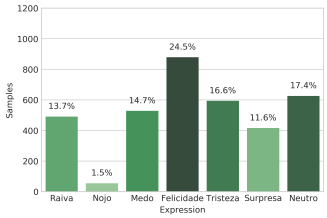

Figure 4. Distribuição *PrivateTest*

Cada exemplo da base dados consiste de imagens de tamanho fixo de tamanho 48x48, em escala de cinza. A geração destas foi realizada de forma automática, sendo assim, cada exemplo consiste de imagens que contenham somente faces, e que estão quase sempre bem centralizadas, ocupando todo, ou quase todo o conteúdo da imagem, conforme 5.

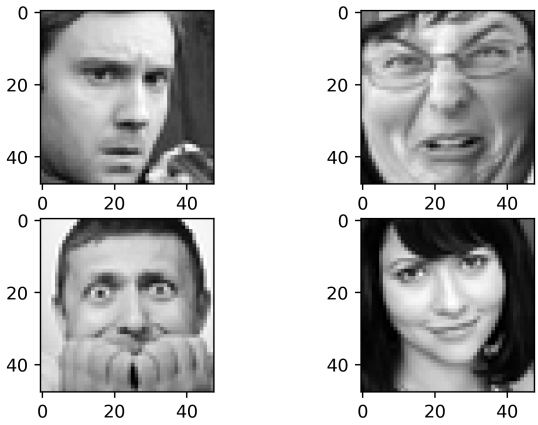

Figure 5. Exemplos da base de dados

Como pode ser observado na Figura 1, a base de dados se mostra bastante desbalanceada. Isto pode ser evidenciado pelo caso extremo da expressão nojo, que está representada por apenas 1.5% de todos os exemplos da base dados. Enquanto, toda as outras expressões estão representadas no mínimo, por quase 10 vezes a quantidade da expressão nojo.

Apesar do grande desbalanceamento da distribuição de exemplos por expressões, observa-se que a proporcionalidade deste desbalanceamento se mostra praticamente constante nas divisões realizadas pelos autores da base de dados, como pode ser visto, nas Figuras 2, 3, 4.

Segundo [], o tamanho de um modelo de CNN está diretamente relacionado ao tamanho da base de dados, onde quanto maior a quantidade de exemplos, maior pode ser o modelo, pois, em caso contrário, não será possível a geração de modelos com maior complexidade e que possuem melhor taxa de generalização. Visto isto, utilizou-se da técnica *Augmentation* para pseudo expansão da da base de dados.

2.2. Construção do modelo

O modelo proposto consiste de um conjunto de *Convolutional Neural Networks* (CNN), onde cada elemento é responsável pela classificação da imagem de entrada em uma das Sete Expressões Faciais Universais [], por meio da produção de vetor de probabilidades. Que consiste na possibilidade de cada uma das expressões universais estarem sendo expressas pela face, na imagem. A determinação da classificação final, não segue as abordagens típicas da literatura, via maioria ou consenso. Pois, estes vetores de probabilidades foram utilizados, de maneira supervisionada, para treinar um modelo do tipo *XGBoost*, que finalmente atribui o rótulo mais adequado à entrada. Tem-se então um *emsemble* de CNN com processo decisório de classificação realizado de maneira não trivial por um modelo de *Machine Learning*.

O objetivo com a união de vários modelos de CNN foi obter classificadores bons de maneira geral em todas as expressões, contudo, estes deveriam se sobressair em determinadas expressões. Com isso, foram gerados sete modelos, um para cada expressão que deveriam ser classificadas, contudo, além dos sete classificadores bons individualmente na classificação de determinada expressão, adicionou-se dois outros, os quais obtiveram melhor desempenho considerando a classificação de todas as expressões, mas não obtiveram melhores resultados em determinada expressão em relação aos classificadores individuais.

Segundo [], o modelo *XGBoost* é um bom modelo para se utilizar na técnica *Boost*, que tem como objetivo obter um modelo *Stronger* baseado nas combinações das respostas de *Weak Learners*. Visto isso, foi considerado cada modelo de CNN, como um *Weak Learner*, apesar de eles não serem fracos, dada a complexidade da tarefa e seus resultado, Tabela 2. E como saída estes forneciam seus resultados, vetores de probabilidade, para o modelo *XGBoost* que tratava de melhor combinar os resultados individuais para se obter um melhor resultado de predição.

As arquiteturas CNN utilizadas na construção do modelo, foram baseadas na arquitetura *VGG-16* [], que consiste de camadas duplas ou triplas de convolução 2D, com *kernel* de 3x3, seguidas de *pooling*, utilizando *MaxPooling*. E como saída tem camadas densas, também chamadas de *Fully Connected* (FC), onde a última utiliza a função de ativação *SoftMax*.

Como função de ativação para as camadas convolucionais e densas, foi utilizada a função *ReLU*, devido ao seu baixo custo computacional, e também possuir derivada constante, o que contribui para o desempenho da função de otimização *adam* []. Segundo [], e análise experimental, a inicialização dos pesos iniciais das camadas convolucionais

que utilizam a função de ativação *ReLu*, com o inicializador de He et al [], possibilita aumento no desempenho, convergência, durante o treinamento dos modelos CNN.

Após a saída de cada camadas convolucionais, foi utilizada normalização em lotes, *Batch Normalization*, pois, segundo [] e análise experimental, os modelos modelos de CNN possuem melhores resultados nas predições, tanto na etapa de treinamento quanto na etapa de generalização.

Algumas das ”regras de ouro” [] para construção de modelos baseados em *Artificial Neural Networks* (ANN) foram utilizadas nas camadas densas, visto que estas são equivalentes. Foram elas: A quantidade de neurônios das cadas ocultas não deve ultrapassar o dobro da quantidade dados da entrada, e a quantidade de neurônios da camada de entrada deve ser a mesma da quantidade de dados da entrada. Vale ressaltar, que as camadas de *FC* em sua maioria não possuem mais do que duas camadas ocultas, devido ao Teorema de aproximação universal [].

Para regularização dos modelos de CNN foi utilizado somente *Dropout*, visto que os regularizadores $l1$ e $l2$, não apresentaram bons resultados durante o período de treinamento, o que tornavam o modelo instável ou com tendências a *underfitting*. Essa tendência foi evidenciada pelo desempenho constante do modelo por algumas dezenas de épocas de treinamento, e ao ser percebido esse comportamento o treinamento foi interrompido.

A arquitetura final dos modelos de CNN podem ser visualizados na Tabela 1. Já a arquitetura utilizada pelo *XGBoost* não pode ser mostrada, devido a esta ter mil árvores classificadoras, o que torna inviável a apresentação desta neste artigo.

Table 1. Arquiteturas utilizadas

1	2	3	4	5	6	7	8	9
Conv2D_(64, 3, 3)	Conv2D_(64, 3, 3)	Conv2D_(64, 3, 3)	Conv2D_(64, 3, 3)	Conv2D_(64, 3, 3)	Conv2D_(64, 3, 3)	Conv2D_(128, 7, 7)	Conv2D_(16, 7, 7)	Conv2D_(16, 7, 7)
BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm
Conv2D_(64, 3, 3)	Conv2D_(64, 3, 3)	Conv2D_(64, 3, 3)	Conv2D_(64, 3, 3)	Conv2D_(64, 3, 3)	Conv2D_(64, 3, 3)	Conv2D_(128, 7, 7)	Conv2D_(16, 7, 7)	Conv2D_(16, 7, 7)
BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm
MaxPool_(3, 3)	MaxPool_(3, 3)	MaxPool_(3, 3)	MaxPool_(3, 3)	MaxPool_(3, 3)	MaxPool_(3, 3)	AvgPool_(3, 3)	AvgPool_(3, 3)	AvgPool_(3, 3)
Dropout_50	Dropout_50	Dropout_50	Dropout_50	Dropout_50	Dropout_50	Dropout_50	Dropout_50	Dropout_50
Conv2D_(128, 3, 3)	Conv2D_(128, 3, 3)	Conv2D_(128, 3, 3)	Conv2D_(128, 3, 3)	Conv2D_(128, 3, 3)	Conv2D_(128, 3, 3)	Conv2D_(128, 5, 5)	Conv2D_(32, 3, 3)	Conv2D_(32, 3, 3)
BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm
Conv2D_(128, 3, 3)	Conv2D_(128, 3, 3)	Conv2D_(128, 3, 3)	Conv2D_(128, 3, 3)	Conv2D_(128, 3, 3)	Conv2D_(128, 3, 3)	Conv2D_(128, 5, 5)	Conv2D_(32, 3, 3)	Conv2D_(32, 3, 3)
BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm
MaxPool_(3, 3)	MaxPool_(3, 3)	MaxPool_(3, 3)	MaxPool_(3, 3)	MaxPool_(3, 3)	MaxPool_(3, 3)	AvgPool_(3, 3)	AvgPool_(3, 3)	AvgPool_(3, 3)
Dropout_50	Dropout_50	Dropout_50	Dropout_50	Dropout_50	Dropout_50	Dropout_50	Dropout_50	Dropout_50
Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	Conv2D_(64, 3, 3)	Conv2D_(64, 3, 3)
BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm
Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	Conv2D_(64, 3, 3)	Conv2D_(64, 3, 3)
BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm	BatchNorm
Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	Conv2D_(256, 3, 3)	MaxPool_(3, 3)	Conv2D_(256, 3, 3)	AvgPool_(3, 3)	AvgPool_(3, 3)	AvgPool_(3, 3)
BatchNorm	BatchNorm	BatchNorm	BatchNorm	Dropout_50	BatchNorm	Dropout_50	Dropout_50	Dropout_50
MaxPool_(3, 3)	MaxPool_(3, 3)	MaxPool_(3, 3)	MaxPool_(3, 3)	Flatten	MaxPool_(3, 3)	Conv2D_(256, 3, 3)	Conv2D_(128, 3, 3)	Conv2D_(128, 3, 3)
Dropout_50	Dropout_50	Dropout_50	Dropout_50	FC_(128)	Dropout_50	BatchNorm	BatchNorm	BatchNorm
Conv2D_(512, 3, 3)	Flatten	Flatten	Flatten	Dropout_20	Flatten	Conv2D_(256, 3, 3)	Conv2D_(128, 3, 3)	Conv2D_(128, 3, 3)
BatchNorm	FC_(128)	FC_(512)	FC_(128)	FC_(64)	FC_(128)	BatchNorm	BatchNorm	BatchNorm
Conv2D_(512, 3, 3)	Dropout_20	Dropout_50	Dropout_20	Dropout_20	Dropout_20	AvgPool_(3, 3)	AvgPool_(3, 3)	AvgPool_(3, 3)
BatchNorm	FC_(64)	FC_(512)	FC_(64)	FC_(32)	FC_(64)	Dropout_50	Dropout_50	Dropout_50
Conv2D_(512, 3, 3)	Dropout_20	Dropout_50	Dropout_20	Dropout_20	Dropout_20	Flatten	Flatten	Flatten
BatchNorm	FC_(7)	FC_(7)	FC_(7)	FC_(7)	FC_(32)	FC_(1048)	FC_(512)	FC_(512)
MaxPool_(3, 3)					Dropout_20	Dropout_20	Dropout_20	Dropout_50
Dropout_50					FC_(7)	FC_(512)	FC_(512)	FC_(512)
Conv2D_(512, 3, 3)						Dropout_20	Dropout_20	Dropout_20
BatchNorm						FC_(7)	FC_(7)	FC_(7)
Conv2D_(512, 3, 3)								
BatchNorm								
Conv2D_(512, 3, 3)								
BatchNorm								
MaxPool_(3, 3)								
Dropout_50								
Flatten								
FC_(512)								
Dropout_20								
FC_(512)								
Dropout_20								
FC_(7)								

2.3. Validação do modelo

A tarefa alvo do modelo final é classificar em qual das Sete Expressões Faciais Universais pertence determinada expressão facial dada como entrada a imagem de uma face. Observando o problema a ser resolvido, percebe-se que este é de natureza classificatória. Na literatura, os problemas dessa natureza tem seu desempenho comumente avaliados via Matriz de Confusão e também da via medida *F*. Onde a medida *F* escolhida foi a *F1 Micro*, devido a natureza desbalanceada da base de dados [].

Para validação do *XGBoost* foi utilizado a validação cruzada estratificada [], para que o desbalanceamento das classes na base de dados fosse replicada nas partições. O número de partições recomendado pela literatura é dez [], contudo, este valor é sugerido para base de dados grandes, o que não reflete a realidade da escolhida. Para a validação deste modelo foi escolhido sete partições, pois, o conjunto *PrivateTest* é aproximadamente $\frac{1}{8}$ da base de dados, o que deixa $\frac{7}{8}$ a serem divididos na validação cruzada. Ressaltando que o conjunto *PrivateTest* foi mantido fixo e utilizado somente como validação final do modelo, para que a comparação com os resultados da competição no *Kaggle* fosse realizada.

Na Figura 6 é possível visualizar a separação original da base de dados, divida em 8 partes de mesmo tamanho, e também é possível observar a separação utilizada na validação cruzada, respectivamente na primeira e segunda linha.

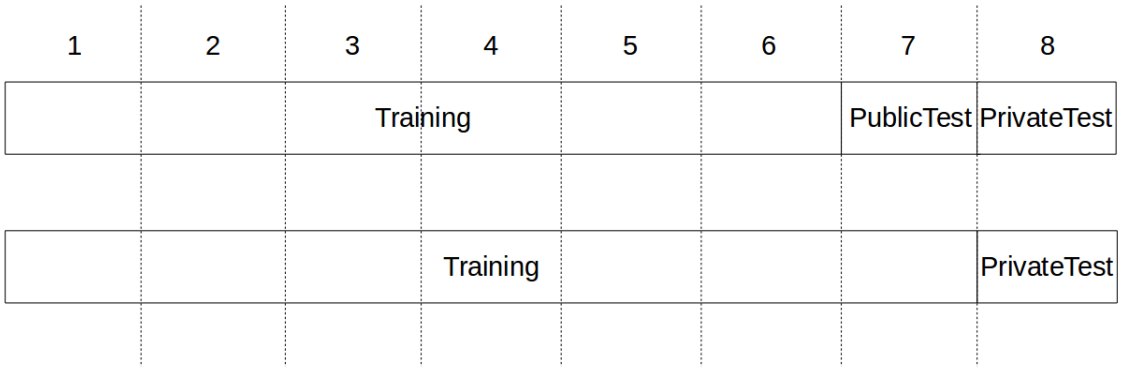

Figure 6. Separação de exemplos da base de dados para Treinamento, Validação e Teste.

3. Resultados e Discussões

A métrica para cálculo de desempenho dos modelos durante a competição no *Kaggle* foi a acurácia, e o melhor resultado obtido na competição foi de 71.161%. Os resultados de acurácia e *F1 Micro* para os modelos de CNN testados, bem com o *Ensemble* obtiveram os mesmos valores, visto isso é apresentado somente o valor da medida *F*. Na Tabela 2 observa-se os resultados para cada modelo de CNN, juntamento do resultado do *Emsemble* das CNN com *XGBoost*.

É observado que o melhor classificador, modelo 7, individual de CNN, obteve 69.49% enquanto que o pior, modelo 9, obteve 62.44%. Ressaltando que cada classificador de CNN usado no *Ensemble* obteve resultados melhores do que os outros, em determinada expressão ou bons resultados em todas as expressões, mas não se sobressaiu

em nenhuma expressão específica. No caso do modelo 9, obteve bons resultados em quase todas as expressões, mas não obteve-se nenhum resultado melhor na classificação de determinada expressão em relação aos outros modelos. Já no caso do modelo 7, obteve-se o melhor resultado de classificação para expressão de surpresa.

Table 2. F1 Micro das Arquiteturas utilizadas

Modelo	F1 Micro
1	0.6898857620507105
2	0.6767901922541097
3	0.6606297018668152
4	0.6798551128448036
5	0.6667595430482028
6	0.6781833379771524
7	0.694901086653664
8	0.6285873502368348
9	0.6244079130677069
ensemble	0.7174700473669546

As matrizes de confusão estão normalizadas. Cada célula foi colorida de acordo com a incidência de elementos contidos nesta antes da normalização(quanto maior a incidência mais escura é a cor da célula). O *grid* de matrizes de confusões da Figura 8 foi gerado a partir da parte *PrivateTest* da base de dados, onde cada matriz corresponde a um modelo de CNN utilizado no *Ensemble*. As classes verdadeiras do exemplo estão representadas pelas linhas, enquanto, as colunas representam as classes preditas pelo *Ensemble*.

Na Figura 7 é apresentado o resultado do modelo utilizando *Ensemble*. Onde o resultado final de desempenho foi de 71.74% de acordo com a métrica *F1 Micro*, ressaltando que seu valor de Acurácia possui o mesmo valor. E com este resultado o *Ensemble* supera, por pouco, o modelo campeão da competição.

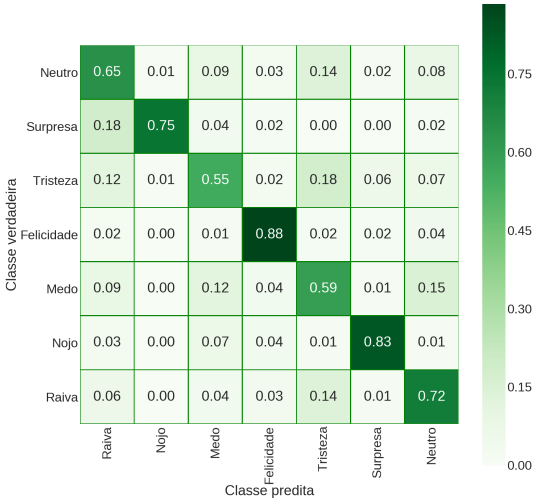

Figure 7. Matriz de Confusão do *Emsemble* (CNN + *XGBoost*)

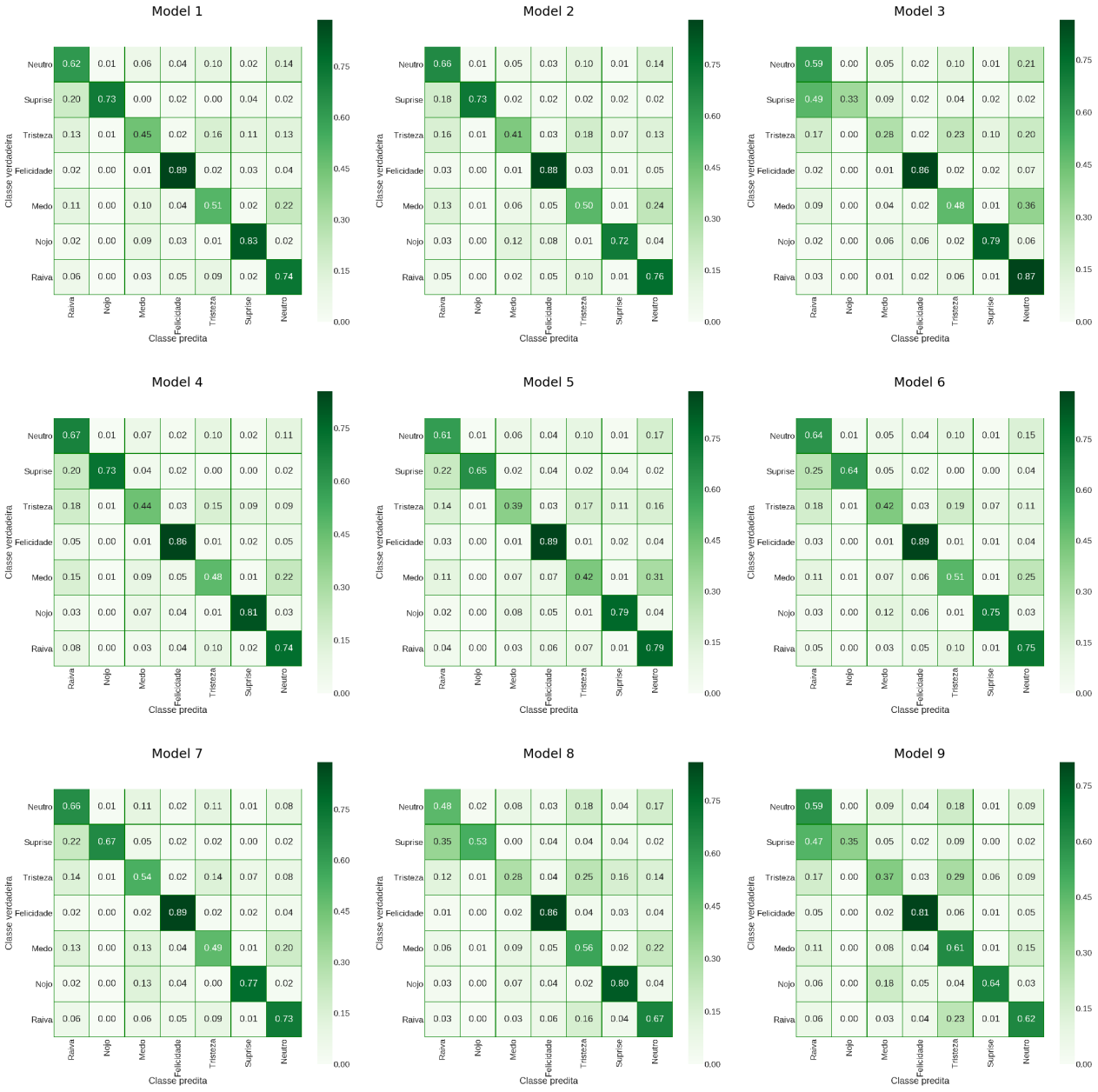

Figure 8. *Grid* de Matrizes de Confusão dos modelos de CNN

4. Conclusão

Texto