

Reconhecimento Automático de Expressões Faciais para Aplicações em Vitrinismo

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

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

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

Abstract. Text

Resumo. Texto

1. Introdução

Texto

Falar aqui das sete expressões faciais universais.

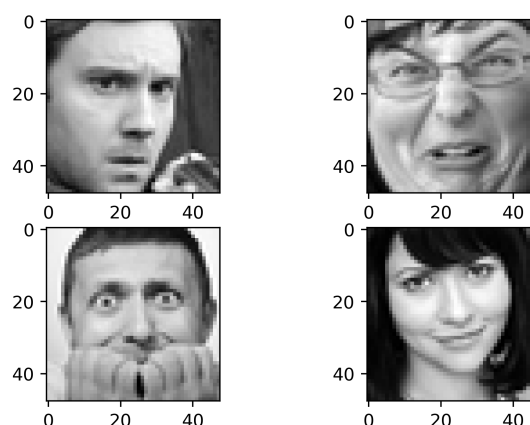
2. Materiais e Métodos

2.1. Dados Experimentais

A base de dados de expressões faciais utilizada para o desenvolvimento deste trabalho é denominada *Facial Expression Recognition Challenge* (FER2013). Esta base contém 35.887 imagens faciais em escala de cinza com dimensões de 48×48 pixels, rotuladas de maneira supervisionada segundo uma das sete expressões faciais universais, conforme amostras ilustradas na Figura 1.

Incluir os rótulos nestes exemplos

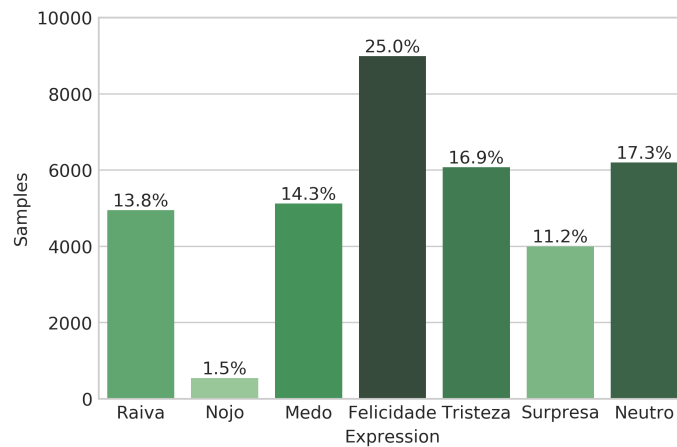
Figura 1: Amostras de imagens faciais e seus respectivos rótulos da base de dados FER2013.



Os exemplos disponíveis na FER2013 se distribuem de maneira heterogênea perante as classes consideradas, conforme ilustra o gráfico da Figura 2. O número de exemplos rotulado com a expressão “nojo”, por exemplo, representam apenas 1.5% do total de exemplos disponíveis. Estas características evidenciam o desbalanceamento do conjunto de dados considerado no tocante à quantidade de amostras por classe.

Rodrigo, veja o gráfico que está na pasta de imagens, intitulado “boxplot-volumemensal”. Este é o padrão para figuras em artigos

Figura 2: Histograma da distribuição de imagens por tipo de expressão facial na base FER2013.



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 base de dados.

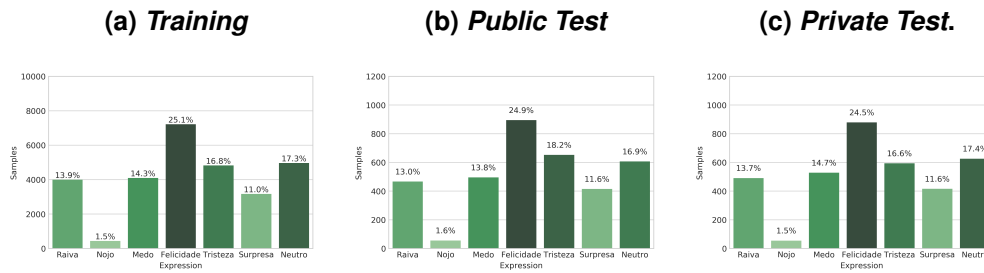
2.2. Critérios de Avaliação

A base de dados em questão será usada para realização de tarefa de classificação segundo o paradigma de aprendizado supervisionado. Nesta tarefa, exemplos de expressões faciais e seus respectivos rótulos serão fornecidos previamente aos modelos de Aprendizado de Máquina para escolha e ajuste de parâmetros e realização do treinamento. Posteriormente, expressões faciais ainda não vistas serão apresentadas e o objetivo será avaliar o desempenho do modelo na classificação destes exemplos, isto é, aferir a respectiva capacidade de generalização.

Obedecendo a uma partição do FER2013 previamente considerada em competições de Visão Computacional [Kaggle 2013], esta base de dados será dividida em 3 partes, sendo: 75% dos exemplos para treinamento (partição *Training*), e 25% dos dados para testes, segundo uma abordagem *holdout* de validação cruzada [Brink et al. 2017]. Do conjunto de testes, metade do mesmo será utilizada livremente (partição *Public Test*), mas a outra metade (partição *Private Test*, cerca de 1/8 da base de dados original) será utilizada para obtenção das métricas de desempenho e comparação dos modelos. Conforme ilustra a Figura 3, as partições preservam a distribuição de amostras por classe na base de dados original.

Duas métricas de desempenho serão utilizadas para comparação dos modelos na realização desta tarefa: a acurácia e o Micro *F-Score*. A acurácia é uma métrica intuitiva que descreve o percentual de acertos do modelo em relação ao total de previsões efetuadas. Embora forneça uma visão geral da capacidade de generalização do modelo, não fornece detalhes acerca dos acertos por classe. Para contornar esta dificuldade, o Micro *F-Score* também será utilizado, pois contempla a média harmônica entre precisão e revocação por classe ao passo que considera as diferentes frequências nas classes do problema [Kubat 2015]. Esta métrica é comumente utilizada em problemas de classificação

Figura 3: Distribuição das classes nas partições adotadas para o conjunto de dados.



com classes desbalanceadas, similar ao cenário considerado no escopo deste trabalho.

Dentre os modelos a serem avaliados, serão elencados como mais aptos para a tarefa considerada aqueles que maximizarem as duas métricas de desempenho para os exemplos pertencentes à partição *Private Test*.

2.3. Proposição de Modelos

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 *ensemble* 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 resultados, 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 previsões, 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 camadas 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.

3. Trabalhos Relacionados

Relatar aqui os trabalhos análogos.

4. Resultados e Discussões

5. 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, juntamente 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.

Tabela 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)								

Tabela 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 5 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 4 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.

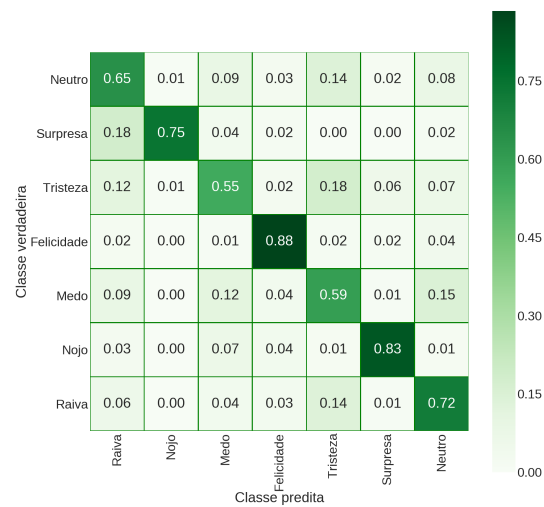


Figura 4: Matriz de Confusão do *Emsemble* (CNN + XGBoost)

6. Considerações Finais

Texto

Referências

- [Brink et al. 2017] Brink, H., Richards, J. W., and Fetherolf, M. (2017). *Real-World Machine Learning*. Manning Publications, Estados Unidos.
- [Kaggle 2013] Kaggle (2013). Challenges in representation learning: Facial expression recognition challenge.
- [Kubat 2015] Kubat, M. (2015). *An Introduction to Machine Learning*. Springer, Estados Unidos.



Figura 5: Grid de Matrizes de Confusão dos modelos de CNN