RECONHECIMENTO DE PARENTESCO UTILIZANDO REDES NEURAIS CONVOLUCIONAIS PROFUNDAS A PARTIR DE FOTOS

ALEXANDRE THUROW BENDER[1]; RICARDO MATSUMURA DE ARAÚJO [2]

[1]*UFPel – atbender@inf.ufpel.edu.br*
[2]*UFPel – ricardo@inf.ufpel.edu.br*

1. INTRODUÇÃO

Nas últimas décadas, modelos computacionais vêm obtendo um notável desempenho nos mais diversos domínios (TESAURO, 1995). A tecnologia de Redes Neurais Convolucionais Profundas, mais comumente referenciadas na literatura como CNNs (*Convolutional Neural Networks*), permitiu que computadores solucionassem problemas de visão computacional que antes exigiam humanos (KRIEZHEVSKY, 2012). O problema apresentado neste trabalho é o reconhecimento de parentesco através de imagens. Esse problema faz parte de um desafio do Kaggle, uma plataforma direcionada para cientistas de dados treinarem e resolverem problemas de predição e aprendizado de máquina.

Esse problema é relevante pois resolvê-lo possibilita identificar traços familiares de forma acessível e sem procedimentos invasivos, o que pode ajudar na construção de árvores genealógicas ou até mesmo ajudar a encontrar filhos e pais perdidos em cidades utilizando câmeras de monitoramento.

A motivação deste trabalho está no fato de que indivíduos relacionados por sangue frequentemente compartilham de alguma característica física facial. Tais características (normalmente chamadas de *features,* na literatura) podem ser identificadas e modeladas utilizando modelos de CNNs.

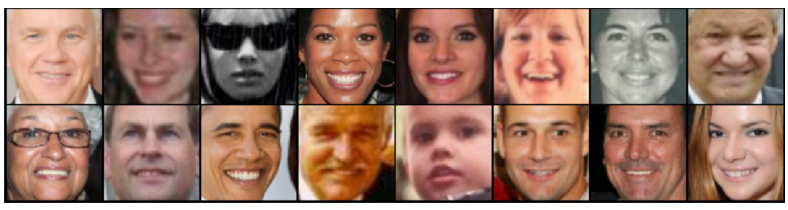
O objetivo consiste em identificar corretamente, dado duas imagens de faces de indivíduos, se essas pessoas possuem parentesco ou não.

A contribuição deste trabalho está na exploração do potencial de arquiteturas de CNNs aplicadas ao domínio do problema de reconhecimento de parentesco utilizando imagens faciais.

2. METODOLOGIA

O dataset do problema em questão contém 18.661 imagens divididas em 1.000 famílias. Foi obtido na plataforma Kaggle, provindo de Families In The Wild (FIW), que é o maior dataset para análise de parentesco atualmente, constituído majoritariamente por imagens disponíveis publicamente de celebridades (ROBINSON et al., 2017).

Figura 1 — Exemplo de Pares de Imagens do Dataset



As imagens do dataset estão divididas em mil famílias e dentro delas constam várias imagens separadas por indivíduo. Também pussui um arquivo CSV indicando os rótulos de parentesco, esse arquivo é necessário pois nem todos os indivíduos em uma família partilham uma relação de parentesco (por exemplo, um pai e uma mãe são relacionados por sangue com suas crianças mas não entre si).

Para treinar a rede utilizou-se 90% do dataset para o conjunto de treino e 10% para o conjunto de validação. Por fazer parte de um desafio do Kaggle, a informação a respeito do conjunto de teste é gerenciada pela plataforma em si.

Quanto à arquitetura do modelo, utilizou-se uma rede siamesa, que é um modelo de rede neural que compartilha seus pesos enquanto trabalha com dois vetores de entrada diferentes, computando vetores de saída comparáveis. Redes siamesas são conhecidas por sua aplicação em problemas de verificação (LeCun, 1994), como por exemplo verificar se a foto de um indivíduo em um documento remete de fato ao portador do documento.

A rede foi testada com algumas variações da AlexNet (limitadas pela memória disponível do ambiente utilizado), onde como entrada são utilizadas duas imagens de tamanho 100x100 pixels e um rótulo indicando se existe ou não parentesco entre os indivíduos.

Cada uma das imagens passa por 5 camadas convolucionais, utilizando ReLU como função de ativação. Essa função propicia a generalização do aprendizado, onde o modelo consegue aprender melhor as *features* presentes nos dados utilizados. O uso de ReLU também reduz *vanishing gradient*, que é um problema onde há a perda do gradiente em modelos com muitas camadas, tornando muito lento o cálculo do erro no treinamento da rede (NAIR et al., 2010). Também foi utilizado *BatchNorm* (SZEGEDY et al., 2015) e *Dropout* (HINTON et al., 2012) em cada uma das camadas. A configuração das camadas de melhor desempenho, entre as testadas, foi 32 x 64 x 64 x 128 x 128, todas elas com tamanho de filtro 3.

Após serem processadas individualmente pelas convoluções, dois vetores de características resultantes são concatenados e passados para uma rede *Fully Connected* de 4 camadas de processamento, sendo estas 2560000 x 200 x 100 x 20 x 2. A conexão do processamento convolucional com o classificador (*Fully Connected*) mostrou-se o maior gargalo de memória do modelo pela grande quantidade de parâmetros.

Para melhorar o desempenho do modelo, foi testada a hipótese de que o formato YCbCr (ao invés do RGB) propiciaria melhor generalização da rede, devido ao fato de que este formato separa o componente luminoso do sinal dos dois componentes de cor. Esse formato costuma ser utilizado em processamento de vídeo pois propicia uma compressão mais eficiente.

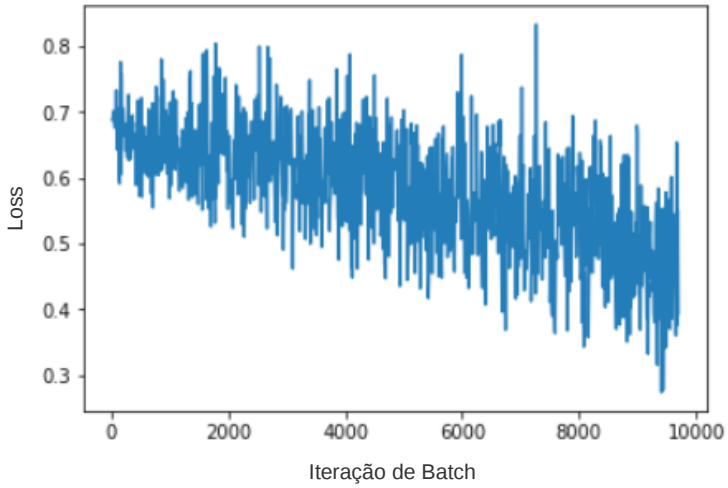
O treinamento do modelo foi realizado por 100 épocas e utilizando *batches* de tamanho 32 no *kernel* disponibilizado pelo Kaggle, utilizando *Cross Entropy Loss* como critério de avaliação e *Stochastic Gradient Descent* como otimizador. Para treinar o modelo, foram utilizados os parâmetros *learning rate* de 0,001 e *momentum* de 0,9.

3. RESULTADOS E DISCUSSÃO

O treinamento durou cerca de uma hora utilizando a GPU disponibilizada pelo Kaggle (Nvidia Tesla P100). No conjunto de validação o modelo obteve até 67% de acurácia e 0,3 de *loss*.

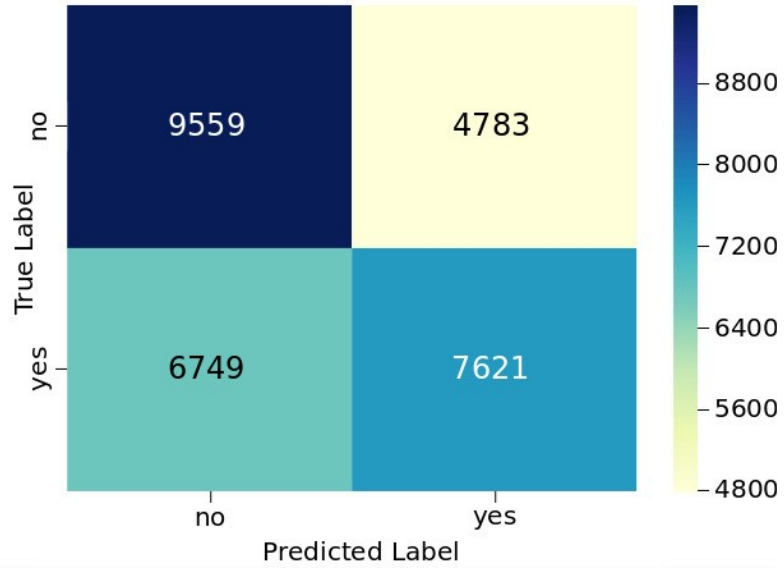
Ao aplicar no conjunto de teste gerenciado pelo Kaggle, o modelo obteve uma acurácia de 63%. Não houve mudanças significativas utilizando o formato YCbCr, quando comparado a RGB. O gráfico de *loss* por iteração de *batch* do sistema denota um decrescimento da *loss*, mesmo que ruidoso. Isto ocorre potencialmente devido ao gráfico ter sido gerado pelo conjunto de validação durante a etapa de treinamento da rede, que é uma boa prática pois facilita observações no comportamento do modelo quando aplicado ao conjunto de validação durante a etapa de treinamento.

Figura 2 — Gráfico de *Loss* por Iteração de *Batch*



A tabela de contigência (ou matriz de confusão) evidencia que o modelo fornece o resultado correto na maior parte dos casos, com as maiores contagens na diagonal principal.

Figura 3 — Tabela de Contigência Após Treinamento



4. CONCLUSÕES

É razoável concluir que o modelo proposto consegue generalizar em algum nível aspectos e características físicas que influenciam na probabilidade de parentesco entre dois indivíduos. Porém, devido às limitações de memória do *kernel* utilizado, o modelo em questão não consegue ser tão representativo e acurado quanto o esperado.

Também conclui-se que a adoção do formato YCbCr não resulta em mudanças significativas nos resultados, ao contrário do esperado, onde o formato YCbCr influenciaria a generalização da rede de forma positiva.

Independente do quão apropriados são os resultados para a utilização do modelo, eles permanecem como uma contribuição para a literatura demonstrando a utilidade e aplicação de modelos convolucionais em problemas de classificação em imagens.

Algumas estratégias futuras que podem potencialmente otimizar o modelo consistem em aumentar o tamanho da arquitetura e treiná-la utilizando um *kernel* diferente que suporte-a; utilizar da técnica de pré treinamento em faces ou transferência de aprendizado empregando modelos que consigam identificar e/ou classificar faces; e ainda aumentar o *dataset* utilizando técnicas de *data augmentation*, principalmente alterando o componente luminoso das imagens.

5. REFERÊNCIAS BIBLIOGRÁFICAS

TESAURO, G. **Temporal difference learning and td-gammon**. Communications of the ACM. 1995.

KRIEZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. **Imagenet classification with deep convolutional neural networks**. In advances in neural information processing systems. 2012.

ROBINSON, J.P.; SHAO, M.; ZHAO, H.; GILLIS, T.; FU, Y. **Recognizing families in the wild: data challenge workshop in conjunction with acm mm 2017**. 2017.

WANG, S.; ROBINSON, J.P.; FU, Y. **Kinship verification on families in the wild with marginalized denoising metric learning**. 2017.

NAIR, V.; HINTON, G.E. **Rectified linear units improve restricted boltzmann machines**. 2010.

BROMLEY, J.; GUYON, I.; LECUN, Y.; SÄCKINGER, E.; SHAH, R. **Signature verification using a siamese time delay neual network**. Advances in Neural Information Processing Systems. 1994.

SZEGEDY, C.; IOFFE, S. **Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift**. 2015.

HINTON, G.E.; SRIVASTAVA, N.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R.R. **Improving neural networks by preventing co-adaptation of feature detectors**. 2012.