Máquina de aprendizagem para identificação de cactos em zonas desérticas

Adner Matheus Andrade Silva
Rodrigo Duarte Xavier da Costa
Daniel Rodrigues de Castro

Resumo
Este relatório pretende descrever a aplicação de técnicas de machine learning, em particular, a Convolutional Neural Network (CNN), para identificação da desertificação e redes hidrográficas subterrâneas. O experimento simula a coleta de dados realizado pelo projeto VIGIA, que, através de drones em regiões áridas e desérticas detecta de cactos, da espécie *neobuxbaumia tetetzo*, determinantes à obtenção de água subterrânea. A partir do desenvolvimento de duas arquiteturas paralelas, variamos seus parâmetros a fim de obter melhores resultados e compará-los com arquiteturas famosas, como a LE-Net5 e VGG 16. Obtendo, ao fim, um resultado de 96% de acurácia. Resultado esse que se demonstrou bastante satisfatório em comparação as arquiteturas premiadas (95% e 98,9%, respectivamente).

1.Introdução

Com a evolução das capacidades de processamento e das metodologias preditivas baseadas em Inteligência Artificial, a utilização de máquinas de aprendizagem vêm se apresentando, cada vez mais, como tendência analítica. Seja para fins biológicos (identificação e classificação de sintomas baseados em imagem), econômicos (análise de crédito e flutuações de mercado) e até ambientais (como será abordado neste trabalho), suas técnicas têm transformado o processamento de dados e inovado a produção de bens e de serviços

Os estudos em Ecologia, como se era esperado, foram incrementados com estas técnicas. O presente trabalho busca explorar e desenvolver uma das renomadas aplicações de IA na área, um projeto patrocinado pelo Consejo Nacional de Ciencia y Tecnologia (CONACYT) denominado de VIGIA: Vigilancia Autónoma de Reservas de la Biósfera. Esta política pública objetiva a identificação e mapeamento de cactos colunares, da espécie *neobuxbaumia tetetzo*, através de drones tripulados, para metrologia da rede hidrológica e avaliação da desertificação e do desmatamento.

Em um primeiro momento, o grupo de pesquisadores (López-Jiménez et al, 2019) do VIGIA coletou e disponibilizou a arquitetura na plataforma Kaggle, além da publicação do artigo que apresenta o estudo de caso e a exploração da arquitetura de redes. Em seguida, o projeto patrocina uma competição no Kaggle, fornecendo a cientista de dados do mundo inteiro a oportunidade de contribuir com o projeto.

Partindo da pesquisa desenvolvida e de todas as arquiteturas envolvidas no projeto VIGIA, este trabalho intenta a análise e desenvolvimento de redes neurais (deep learning) para aplicação no dataset. CNNs serão investigadas, assim como a coerência de medidas de desempenho e limitações das soluções aqui apresentadas.

2. Revisão da literatura

2.1 Avaliação de desempenho

Quando se trabalha em uma área com diversos valores e mudanças sensíveis é importante utilizar de métricas respeitadas para validar seus resultados para quantificar seu poder discriminativo do que é um bom modelo ou não para a tomada de decisão. Partindo desse pressuposto é interessante levantar uma breve revisão de como funciona e o que é as principais medidas de desempenho encontradas na literatura.

2.1.1 Matriz de confusão

A matriz de confusão é uma importante ferramenta por sua velocidade e sua facilidade visual de entender a qualidade do modelo, se destoando de um valor único (uma porcentagem) e indo para uma área mais visual. Sua forma de atuação acontece com um conjunto de dados chamados de teste ao método classificação e verificando os resultados que o modelo escolheu em comparação com sua classificação real. O eixo vertical corresponde às classes reais e o eixo horizontal as classes do modelo. A matriz confusão tem sua relevância em ser primeira tratada aqui pois ela dá embasamento para as outras métricas.

Figura 01: Matriz confusão contendo duas classes

Fonte:http://crsouza.com/2009/07/13/analise-de-poder-discriminativo-atraves-de-curvas-roc/

2.1.2 Acurácia e Loss

Definida como a divisão do total de acertos pelo total de dados no conjunto, é outra importante medida de desempenho é a acurácia,, porém, mesmo que essa métrica seja amplamente usada nos modelos ela é suscetível a uma falsa interpretação dos resultados pois em sua concepção não leva em consideração o desbalanceamento do conjunto de dados, podendo ter mais negativos e ao classificar muitos negativos e poucos positivos o modelo não generaliza bem.

$$Accuracy = \frac{VP + VN}{P + N}$$

A função de perda (loss function) é uma importante função de desempenho da área de machine learning que tem como diferencial mapear as decisões seguintes associadas ao custos de desempenho do sistema. A forma como ela atua é comparada à uma descida de uma montanha procurando o caminho que lhe causa menos energia. Há diversos tipos de function loss. Seu uso em machine learning se dá de forma que minimiza o erro de cada exemplo de treinamento durante o processo de

aprendizado. Isso é feito usando algumas estratégias de otimização, como a gradient descent, e esse erro vem da função de perda.

2.2 Data Augmentation e Fit_Generator

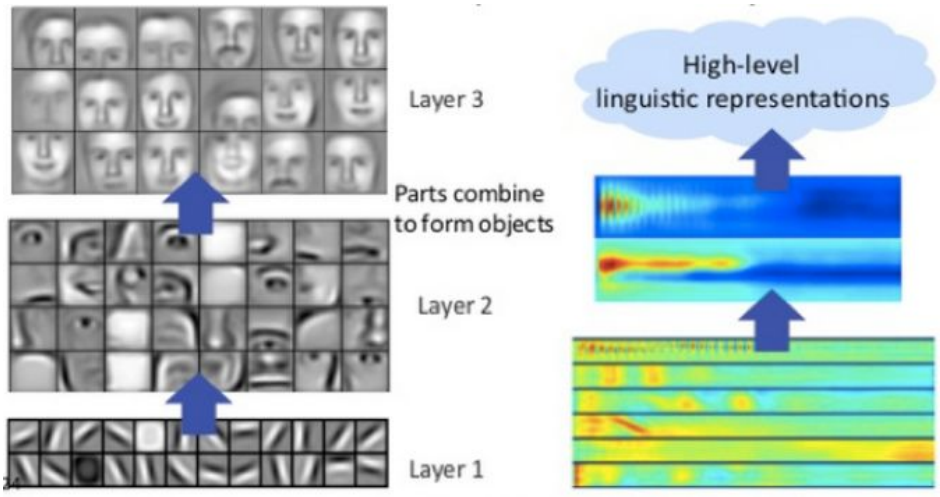
Data augmentation é uma técnica utilizada para aumentar o banco de dados, fazendo com que as imagens assumem diferentes ângulos, texturas e cores, retirando também algumas partes das mesmas e adicionando máscaras. Dessa forma, a variabilidade das imagens analisadas aumenta de forma significativa, sendo essencial para a generalização do modelo.

O fit_generator é uma função da biblioteca keras que randomiza as imagens durante o aprendizado em redes neurais, sua parametrização será fundamental para a medida de desempenho do aprendizado e para efetividade do backpropagation. A cada step (uma unidade de batch) o fit generator gera novas imagens e as randomiza nos outros batchs, sendo assim, durante o treino, o modelo testa sua capacidade tanto de generalização (imagens ainda não inseridas) quanto de identificação.

2.3. Convolutional Neural Network (CNN) e Rede Neural

Os algoritmos de deep learning são um subconjunto dos métodos de feature learning (Nielsen, 2015). Tais métodos de feature learning tem por objetivo escolher e analisar o peso de cada feature no resultado dos modelos. Os algoritmos de deep learning são caracterizados por conter múltiplas camadas de features entre as camadas de entrada e de saída trazendo através da passagem de cada camada uma noção mais abstrata das features para a camada seguinte. Na classificação de imagens, por exemplo, a primeira camada pode extrair curvas e repassar essa informação para a camada posterior. A segunda camada por então combinar algumas dessas curvas formando partes de objetos. Já a terceira camada compõe objetos mais complexos que então serão usados para classificação, utilizando pesos para orientar o processo (Gutemberg, 2017).

Figura 02: Representação simples de uma rede neural

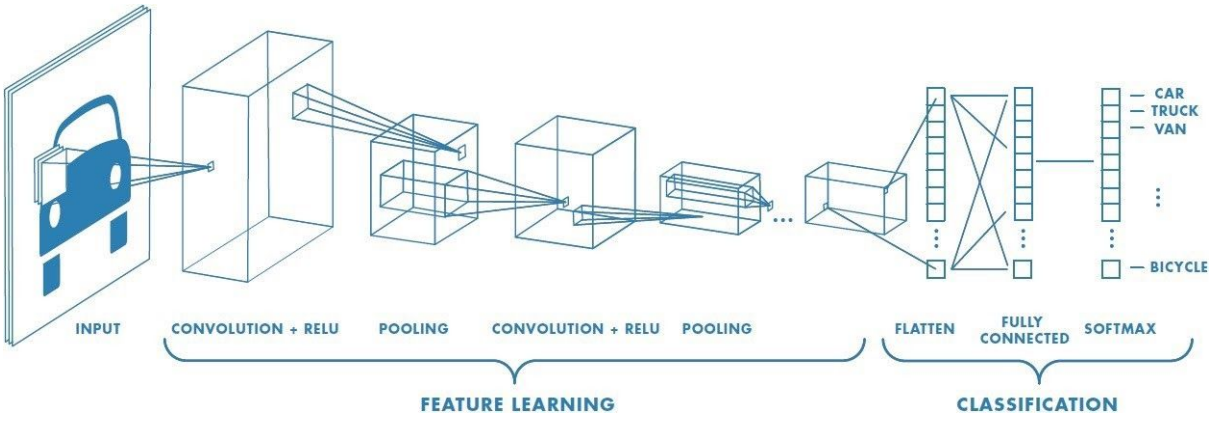


Fonte: https://skymind.ai/wiki/neural-network

Uma CNN é um método do conjunto de deep learning que tenta extrair certas características através de filtros de convolução, cujo significado matemático é o tratamento de uma matriz por outra utilizando um kernel. Sua principal função é processar dados em múltiplas dimensões, como por exemplo imagens. Uma imagem armazenada no padrão RGB (red, green, blue) possui altura, largura e profundidade, considerando a profundidade como sendo as 3 camadas de cor utilizadas na

representação (Cipollini, 2015). Uma CNN é composta originalmente por unidades de convolução, unidades de pooling e uma unidade de classificação.
Na unidade de convolução é passada uma janela com pesos definidos passando pela imagem capturando certas características, analisando a profundidade de cada imagem gerando um resultado chamado de mapa de características. A unidade de pooling reduz a dimensão espacial da entrada de forma similar a unidade convolução escolhendo um representante da vizinha no processo. A unidade de escolha é a última camada e é um classificador totalmente conectado. Esse classificador irá receber como entrada o vetor de características extraídas pelas camadas anteriores e não a imagem original. Segue abaixo uma representação simples de uma rede convolucional.

Figura 03: Representação de uma rede neural convolucional



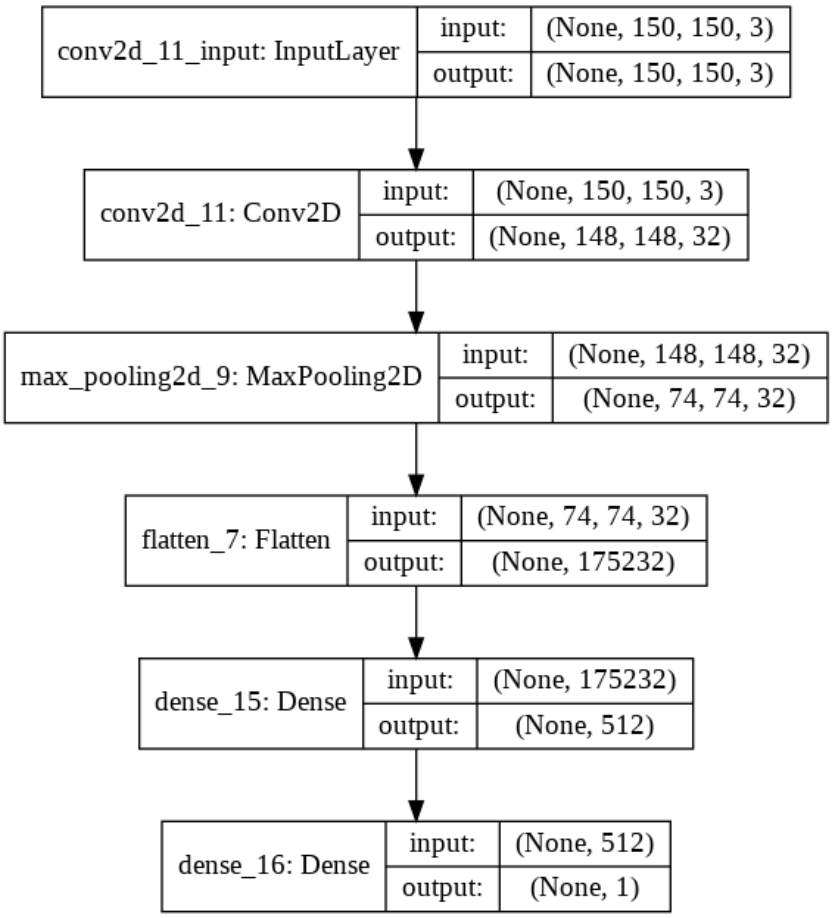
Fonte: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

3. Metodologia

Foi utilizado Data augmentation, uma vez que o dataset utilizado foi desenvolvido para um uso específico da avaliação de desertificação e desmatamento de determinadas áreas, e possuir também muito mais dados que possuem cactos do que o contrário, sendo cerca de 3 vezes maior a amostra de imagens que possuem cactos. O tratamento dos dados necessário não se demonstrou complexo, devido ao formato que as imagens se encontram, sendo mais simplificado o tratamento sem a necessidade de transformar o formato das imagens utilizadas.

Em geral, duas arquiteturas bases foram utilizadas neste trabalho. A primeira, aqui denominada como “default” é representada pela figura 04 e conta com os elementos mais básicos de uma CNN. Nela, testamos o impacto individual de parâmetros como o padding, funções de ativação, aderência de novas camadas de CNN, aderência de novas camadas de neurônios pós-flatten, otimizadores.

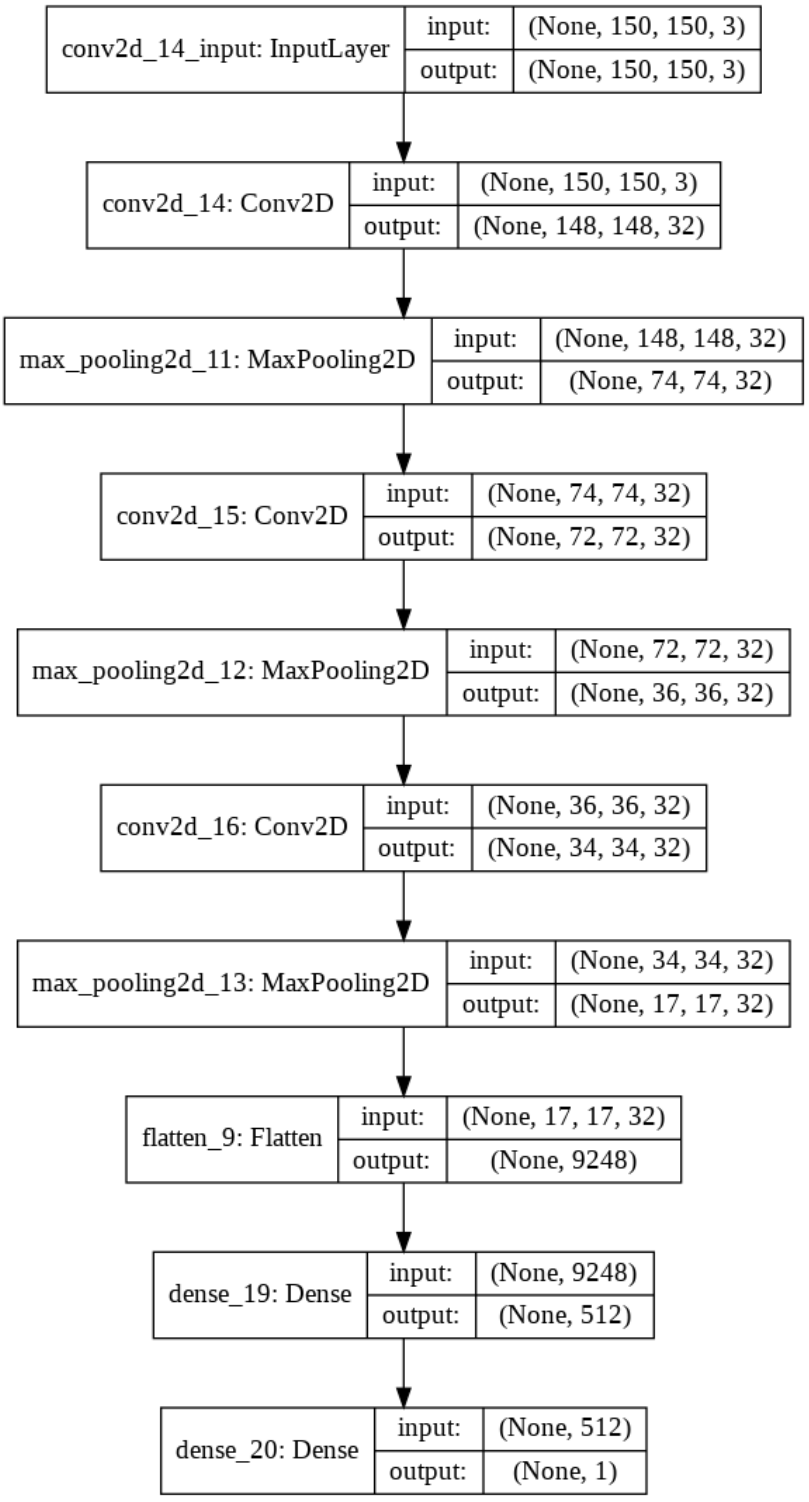
Figura 04: Arquitetura default



Fonte: Autores

Em seguida, construímos uma arquitetura que denominamos de “complexa”, esta, por sua vez, repetia o ordenamento da default no e os parâmetros default de nossa primeira arquitetura. Essa Rede neural está representada pela figura 05 e seu objetivo inicial estava, a priori, no teste da relevância da complexidade. Inferências de aprendizado durante o percurso acrescentaram bastante no desenvolvimento de nossa arquitetura final, que nada mais foi do que uma variação da arquitetura complexa gerada a partir de uma criteriosa análise de sensibilidade dos parâmetros.

Figura 05: Arquitetura complexa



Fonte: Autores

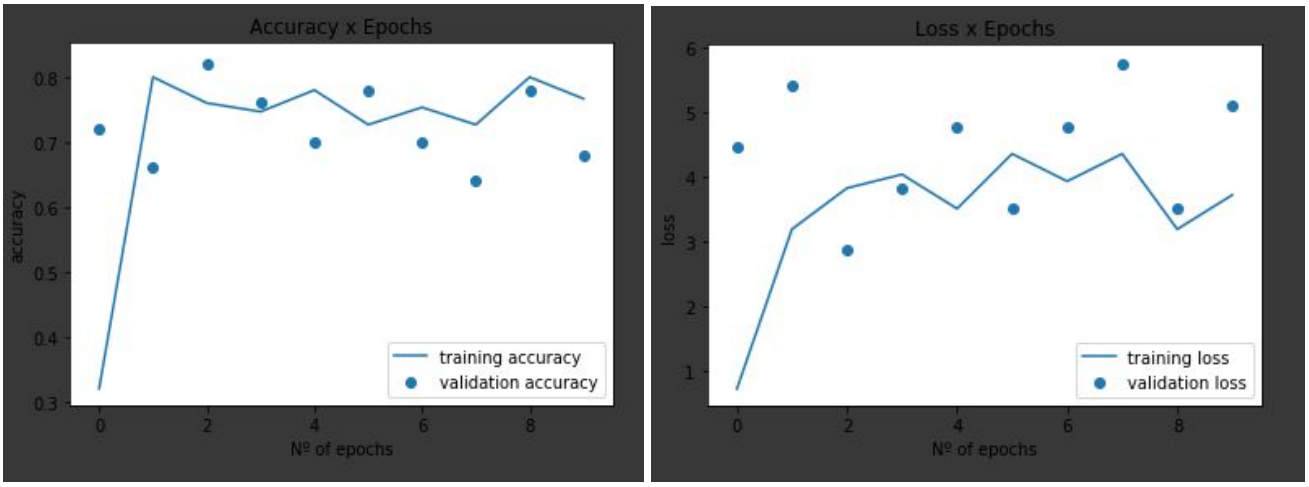
No decorrer deste trabalho mostraremos os principais avanços e análise dos resultados, seguido de uma comparação crítica de outras arquiteturas famosas também aplicadas no dataset.

4. Estudo de caso

O trabalho se desenvolveu a partir da experimentação contínua e avaliação de dois tipos gráfico, uma série temporal das épocas em relação à acurácia e outra envolvendo a loss. Ambas intercalando o desempenho de treino e a da validação.

Iniciamos com a arquitetura Default e tivemos o desempenho descrito na figura 06. Importante fazer uma ressalva em relação aos parâmetros utilizados, neste primeiro teste, a partir de nossos estudos sobre o upload das imagens, decidimos por realizar a data augmentation dentro da rede geral, inserindo novas imagens a cada step. A implicação deste método resulta numa estabilização rápida do aprendizado, onde a maioria dos exemplos, inclusive não ultrapassam de 15 epochs. A diferenciação aqui, se encontra na quantidade de steps dentro de uma época, onde cada um terá sua própria acurácia e a do último step será a do epoch.

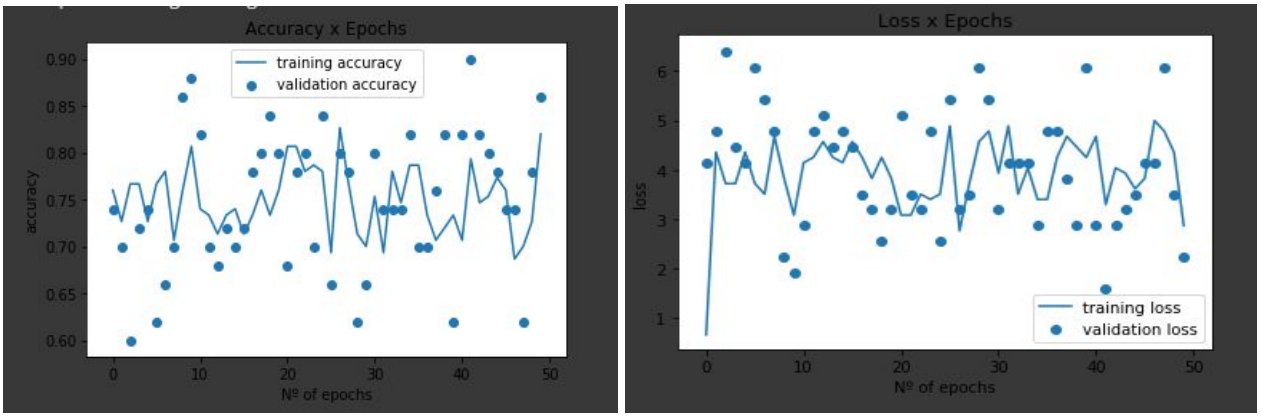
Figura 06: Desempenho da arquitetura default



Fonte: Autores

Para validação dos pressupostos, aplicamos a mesma arquitetura para um número maior de epochs (figura 07), cujo tempo de processamento foi superior a 1 hora. É notável também a variância desde as primeira epochs, onde as validations variam entre 60 e 90%, mas sem tendências.

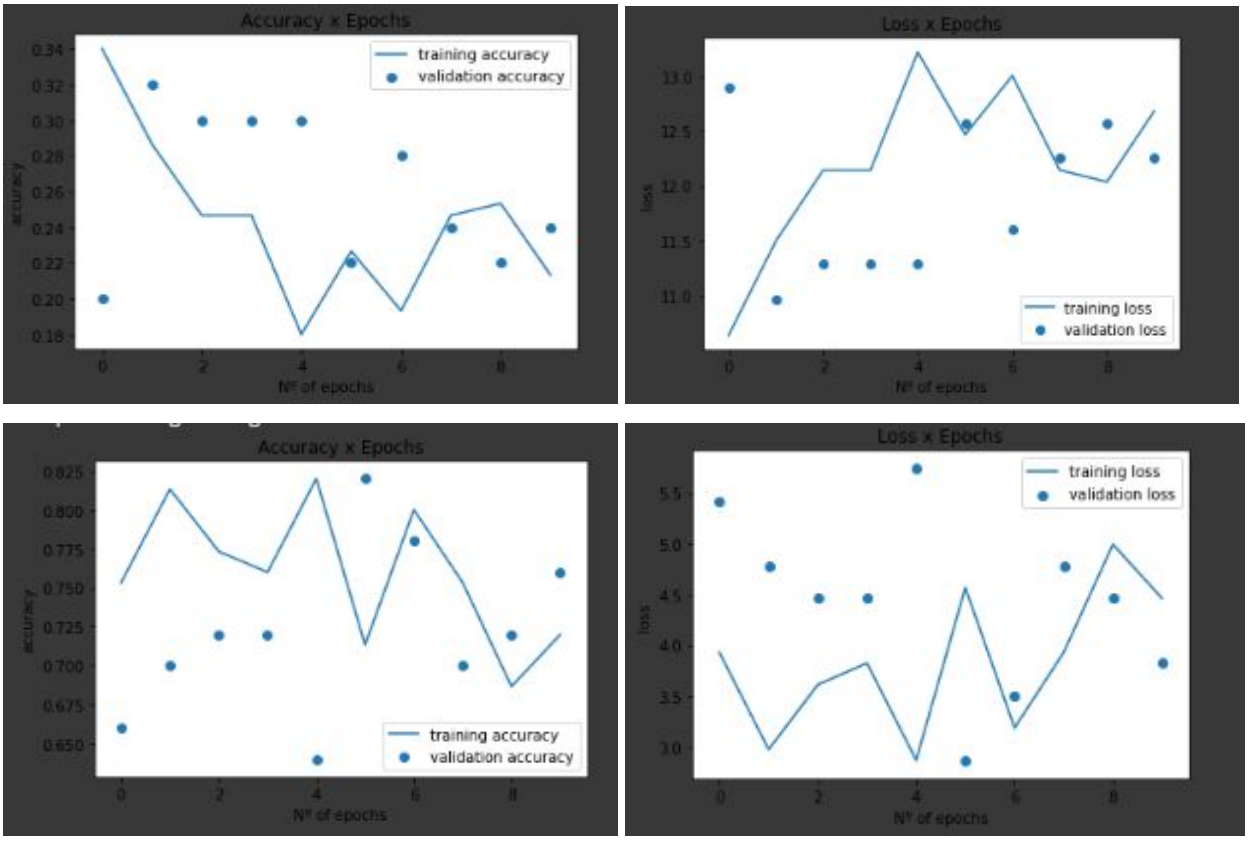
Figura 07: Desempenho da arquitetura default com 50 epochs



Fonte: Autores

Em seguida testamos a variação de parâmetros que julgávamos relevantes ao aprendizado, como a função de ativação na camada de neurônio finais. As duas primeiras testamos o tanh e em seguida a softmax (figura 08), o resultado pode ser entendido a partir da saturação e da loss binary_crossentropy. O desempenho da Loss é até próximo ao da sigmoid, entretanto o avanço das epoch indica um loss levemente crescente e consequente uma diminuição no balanço do treino.

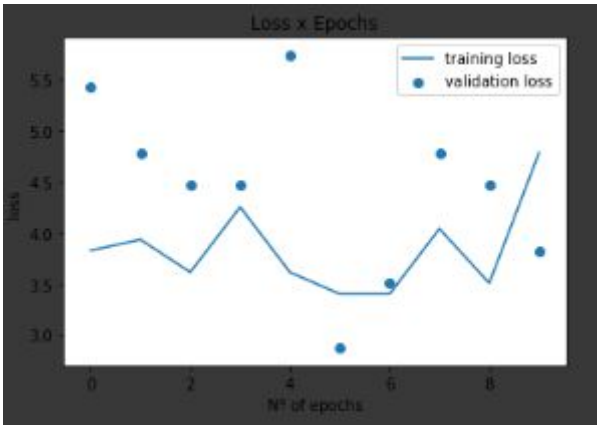
Figura 08: Desempenho da arquitetura default com última camada neural tanh e softmax



Fonte: Autores

Alterações no padding, em outras funções de ativação e em algumas ordens também foram feitas, mas quase sempre inviabilizaram o aprendizado ou não trouxeram grandes ganhos. Após essa fase da experimentação começamos a investigar a relevância da complexidade no aprendizado, tivemos como primeiro desempenho o exposto pelo gráfico da loss na figura 09.

Figura 9: Loss da primeira aplicação de arquitetura complexa



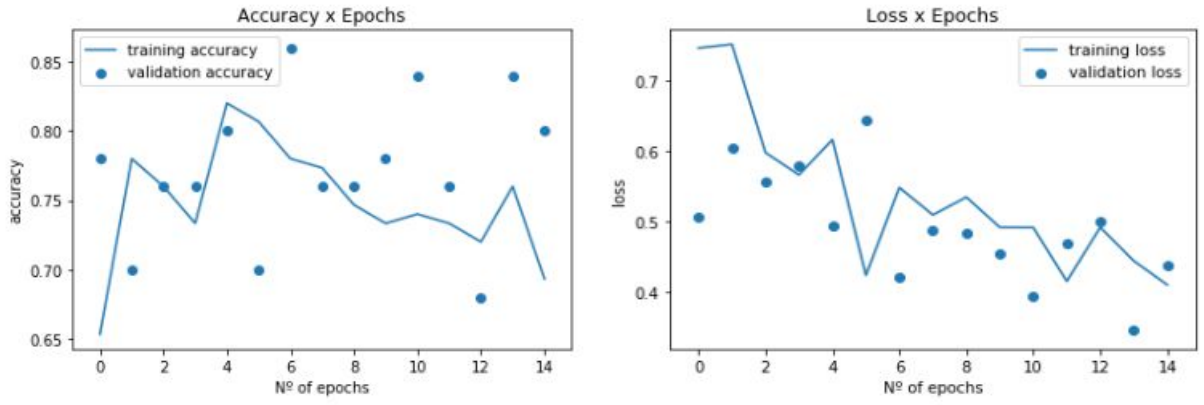
Fonte: Autores

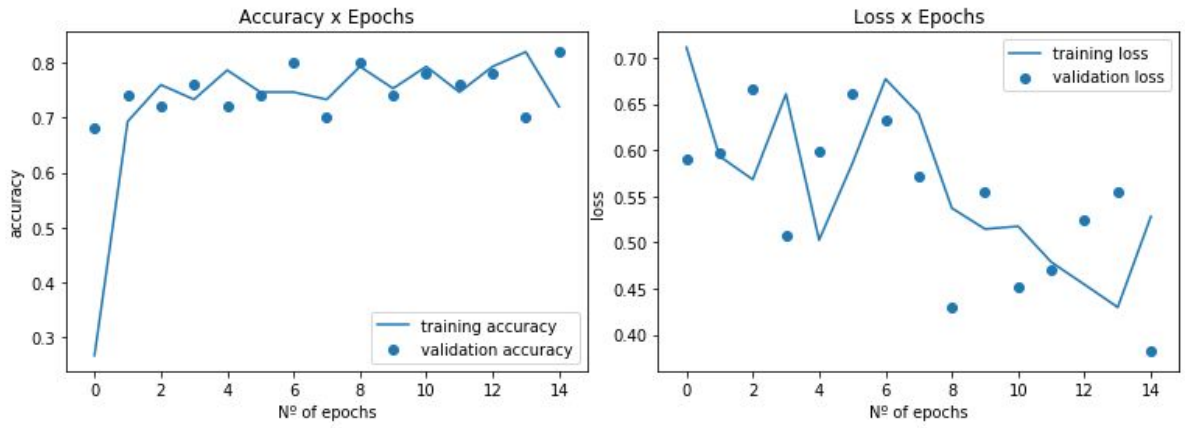
A partir da arquitetura complexa seguimos por duas frentes, a primeira preocupou-se com a reavaliação dos paddings e otimizadores, a segunda, reviu os parâmetros do fit_generator e fez algumas descobertas quanto a arquitetura.

4.1 Paddings e otimizadores

Ainda na experimentação, realizamos a troca do otimizador de adam para rmsprop é revelada uma evolução mais consistente (tem acurácia tendenciando) dos valores na acurácia e na função loss ao longo das épocas na arquitetura complexa, porém seus valores finais são parecidos, como é percebido na figura 10.

Figura 10: Gráficos da acurácia e loss x época na mudança do otimizado adam (em cima) para o rmsprop (em baixo)

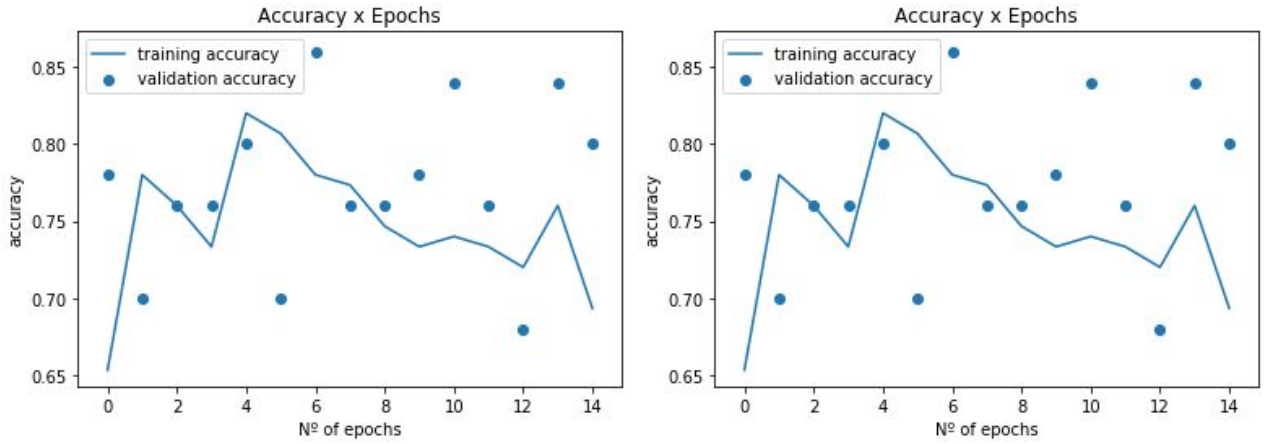




Fonte: Autores

Outro importante insight obtido através dos parâmetros da CNN é o da utilização do padding para analisar melhor as bordas das imagens, colocando 1 dimensão na esquerda, direita, em cima e em baixo das imagens a cada camada convolucional, porém não houve uma súbita melhora como pode ser visto na imagem abaixo:

Figura 11: Mudança na utilização do padding



Fonte: Autores

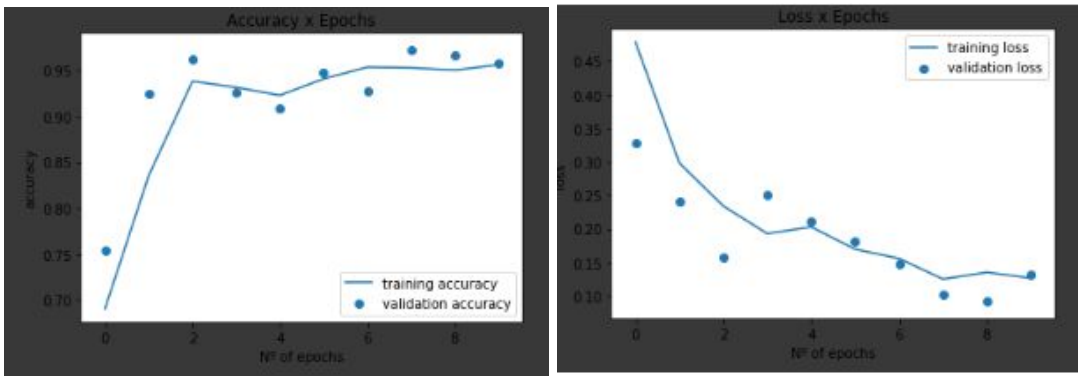
4.2 Steps por epoch e validation_step

Aqui as mudanças foram mais significativas. Neste momento entendemos com mais afinco a aplicabilidade do fit_generator tanto para a separação dos conjuntos de validation e treino quanto para a randomização orientada no data augmentation. Em suma, nas outras etapas de experimentação, cometemos um erro por enviar samples completa, sem unidades de batches, involuntariamente.

Em termos práticos, com a arquitetura completa, enviamos todo o conjunto de dados separados aleatoriamente pelo fit_generator e a cada época concluída, criaríamos um novo conjunto de dados a serem avaliados e a serem re-inseridos no conjunto de dados. Esta medida, obviamente, não melhorava o desempenho e tornava a função improdutiva.

Quando aumentamos o número de steps e o de validation_steps para 10, percebemos a grande diferença na acurácia (figura 12). A partir desses parâmetros podemos observar o seguinte comportamento: o conjunto de dados, já separado randomicamente, era dividido em 10 subconjuntos (10 steps) e que, a cada step, gera e avalia mais randomização de imagens que são re-inseridas no conjunto de dados a adentrarem na próxima época.

Figura 12: Melhor desempenho registrado



Fonte: Autores

Sendo assim, testamos a generalização e a identificação do modelo, evitando fenômenos de underfitting e overfitting e maximizando a acurácia (a final é 96%, entretanto tivemos máximos locais de até aproximadamente 98%)

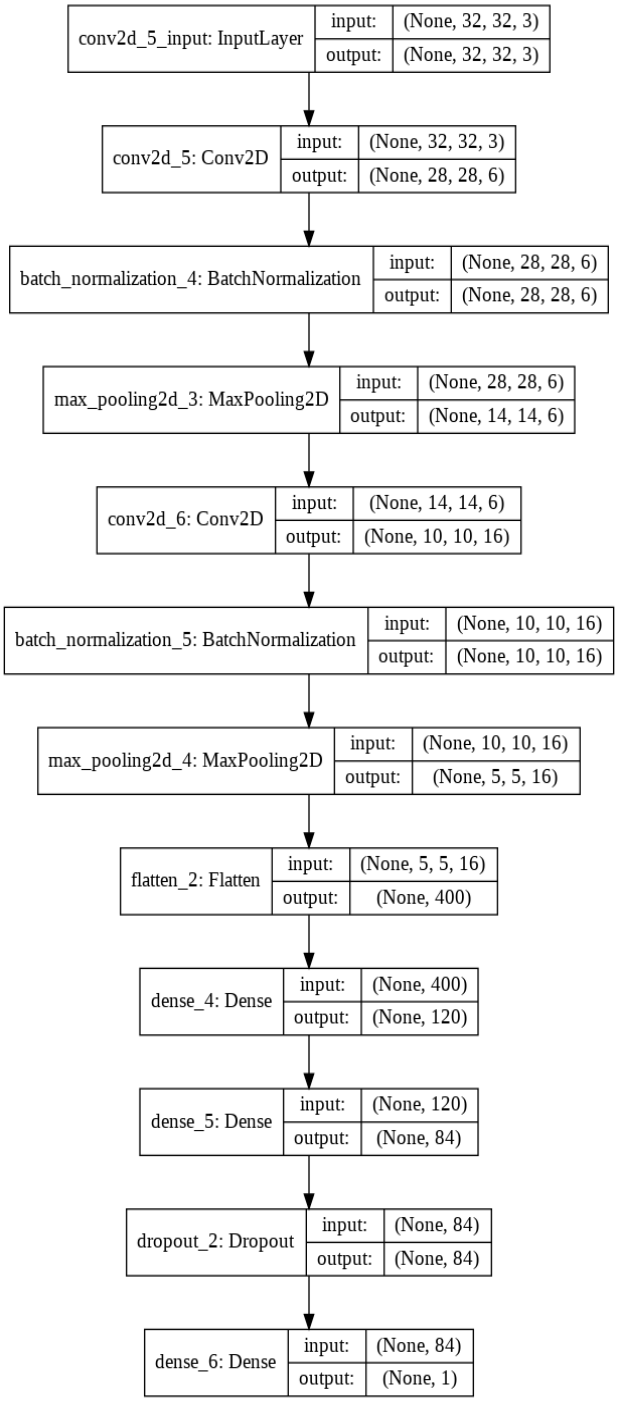
5. Benchmarking de arquiteturas

Outros resultados, suas respectivas arquiteturas e parâmetros, obtidos pela comunidade do Kaggle estão expostos a seguir.

- **LE-Net 5**

Arquitetura utilizada pelo projeto VIGIA (Vigilancia Autónoma de Reservas de la Biósfera) financiado pelo CONACYT (Consejo Nacional de Ciencia y Tecnologia). Fazendo uso da função perda Entropia cruzada binária, Adam como otimizador e acurácia para medição de desempenho, obtendo resultado de 95%.

Figura 13: Arquitetura LE-Net 5



Fonte: https://www.kaggle.com/shahules/getting-started-with-cnn-and-vgg16

- **VGG 16**

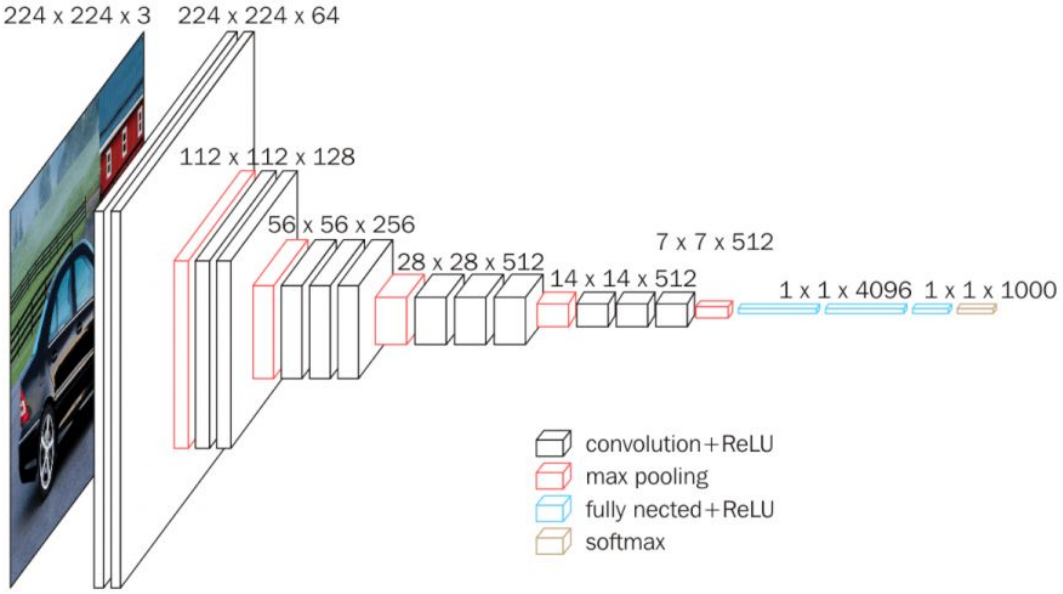
Arquitetura de CNN com 20 camadas algumas das quais sendo seguidas por camadas de MaxPooling e em seguida quatro camadas totalmente conectadas finalizando com um classificador softmax de 1000 vias como representado na Figura 14. Fazendo uso da função perda Entropia cruzada binária, Adam como otimizador e acurácia para medição de desempenho, obtendo um resultado de 98,9%.

Quadro 1: Arquitetura VGG 16

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Fonte: https://engmrk.com/vgg16-implementation-using-keras/

Figura 14: Representação da VGG 16



Fonte: https://neurohive.io/en/popular-networks/vgg16/

6. Limitações

O modelo apresenta limitações envolvendo o dataset pelo qual o modelo foi treinado, tanto em relação a baixa qualidade gráfica das imagens, obtidas sob uma mesmo perspectiva, como na falta de variabilidade da localização de obtenção das mesmas, o que torna o dataset restrito a localidades com características similares. Além disso, o fato do modelo ser binário resulta em baixa interpretabilidade, não sendo levado em consideração a quantidade de cactos que existe na imagem capturada,

dificultando a mensuração do quão forte é a desertificação e a presença de água subterrânea na região, informando apenas se há ou não.

7. Considerações finais

O desempenho do modelo construindo se torna comparável às outras arquiteturas tomadas como referência, LE-Net 5 e VGG 16, caracterizando então um bom modelo de identificação dos cactos. Podendo, dessa forma, ser utilizado para a construção das redes hidrográficas subterrâneas em biomas parecidos aos desses. Já o uso de conceitos como o número de steps, steps validation e a função de ativação sigmoid foram de importante relevância para o desempenho do modelo. Outros insights relevantes se encontram no uso de padding e de outras funções ativações que se mostraram de pequena relevância para as métricas usadas.

Referências

López-Jiménez, E., **Columnar cactus recognition in aerial images using a deep learning approach**, Elsevier B.V., 2019.

A Beginner's Guide to Neural Networks and Deep Learning, Key Concepts of Deep Neural Networks. Disponível em: <https://skymind.ai/wiki/neural-network>. Acesso em: 27/11/2019.

VIGIA: Vigilancia Autónoma de Reservas de la Biósfera. Disponível em: <https://jivg.org/research-projects/vigia/>. Acesso em: 26/11/2019.

DEEP NEURAL NETWORKS HELP US READ YOUR MIND. Disponível em: <https://neuwritesd.org/2015/10/22/deep-neural-networks-help-us-read-your-mind/>. Acesso em: 28/11/2019.

Building powerful image classification models using very little data/ Disponível em: <http://deeplearning.lipingyang.org/wp-content/uploads/2016/12/Building-powerful-image-classification-models-using-very-little-data.pdf>. Acesso em 27/11/2019.

R. Hecht-Nielsen, **Theory of the backpropagation neural network**, 7553, pp. 436-444, 285 2015.

Marques, V., Avaliação do desempenho das redes neurais convolucionais na detecção de ovos de esquistossomose, UFPE, 2017.

Convolutional Neural Networks (CNNs / ConvNets). Disponível em: <http://cs231n.github.io/convolutional-networks/#overview>. Acesso em: 27/11/2019.

WHAT IS THE DIFFERENCE BETWEEN STEP, BATCH SIZE, EPOCH, ITERATION ? MACHINE LEARNING TERMINOLOGY. Disponível em: <https://tolotra.com/2018/07/25/what-is-the-difference-between-step-batch-size-epoch-iteration-machine-learning-terminology/>. Acesso em: 28/11/2019

Tensorflow: train dataset by epochs or steps?. Disponível em: <https://medium.com/@linda0511ny/tensorflow-train-dataset-by-epochs-or-steps-3839705f307d>. Acesso em: 27/11/2019