Utilizando Deep Learning para predizer tipos de plantas

Victor N. Rebli
Universidade Federal do Espirito Santo

***Resumo* — Esse relatório apresenta as técnicas e consequentemente os resultados obtidos na utilização de deep learning por meio do framework keras em uma competição do do site kaggle, em qual o objetivo era reconhecer 99 tipos de espécies de plantas diferentes.**

1. Introdução

O kaggle é um site de competições de modelagem preditiva em que as empresas, ONGs, institutos de pesquisas e outros tipos de organizações disponibilizam seus dados e analistas de dados de diversas partes do mundo competem entrem si em busca do melhor resultado, as vezes resultando em prêmio.

Para esse trabalho foi escolhida a base de dados **leaf classification**, onde o objetivo principal da competição era usar as imagens binárias de folhas de plantas para identificar 99 species de plantas, e como as folhas devido ao seu volume, prevalência e características únicas, são um efetivo meior de diferenciar plantas.

A Base de dados consiste em 1584 imagens, sendo 16 amostras de cada uma das 99 espécies, além de três característica terem sido providos para cada imagem: Um descritor contíguo de forma, um histograma de textura interior e um histograma de margem de escala fina. O conjunto de treino separado pela equipe do kaggle consiste em 990 imagens e o conjunto de teste que será utilizado para submeter a predição do modelo contêm 594 imagens.

A métrica de avaliação escolhida pelo kaggle é o **multiclass loss**, e o uso do dessa métrica permite que durante o período de treinamento o extremo otimismo ou pessimismo seja punido de maneira extrema, de tal maneira que caso o modelo faça uma predição de extrema confiança, como 100% para um classe e 0% para outras, e essa predição esteja errada, nesse caso o erro gerado será muito grande e a correção será proporcional a isso.

A função **multiclass loss** é definido na figura 1

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

Figura 1

onde N é o número de exemplos e M é o número de classes e y_{ij} é uma váriavel binária indicando se a class j estava correta para o exemplo analisado i.

Para esse trabalho foi utilizado a ferramenta keras[1], que é uma biblioteca de alto-nível, escrita em python capaz de utilizar como backend os frameworks tensorflow[2] e theano[3], abstraindo os conceitos dos mesmos e facilitando a escrita do código. Para esse trabalho como foi utilizado o tensorflow como backend.

2. Metodologia

Foi feito um pré-processamento nos dados, como a transformação dos labels(rótulos) do conjunto de treinamento que estão em formato nominal para o formato numérico e os dados de treinamento e de testes foram normalizados entre 0 e 1.

A arquitetura final escolhida para realizar o treinamento dos dados é definida na figura 2.

Como pode-se notar, uma rede neural(deep learning) composta de 5 camadas : são 192 entradas totalmente conectados na primeira camada contendo 1024 neurônios, seguido de um processo de normalização dos dados e aplicação da função de ativação relu(rectifier activation function), sendo seguido novamente por uma segunda camada contendo 1024 neurônios, aplicando a mesma normalização e função de ativação e também o dropout para tornar a rede resiliente contra overfitting; depois disso existe mais duas camadas contendo respectivamente 1024 e 512 neurônios até a ultima camada contendo os 99 neurônios de saída, sendo seus resultados expressos em softmax.

Várias alterações na arquitetura da rede neural foi feita para obter uma melhor pontuação final no kaggle, como a diminuição de neurônios nas camadas, a escolha da função de ativação sigmoid, a alteração do batch size, a inclusão de

mais camadas na arquitetura, o aumento do número de épocas para o treinamento. Todas essas alterações não trouxe aumento significativo na pontuação final, mas porém não houve pioras significativas.
Durante o treinamento foi utilizado o cross-validation com k-folds = 10, com o objetivo de validar durante o treinamento a ocorrência de overffiting, e depois foi executado novamente um modelo final, onde os dados foram repartidos em 80% para treinamento e 20% para validação. E finalizando, o modelo final foi executado no conjunto de testes disponibilizado pela kaggle e submetido o resultado.

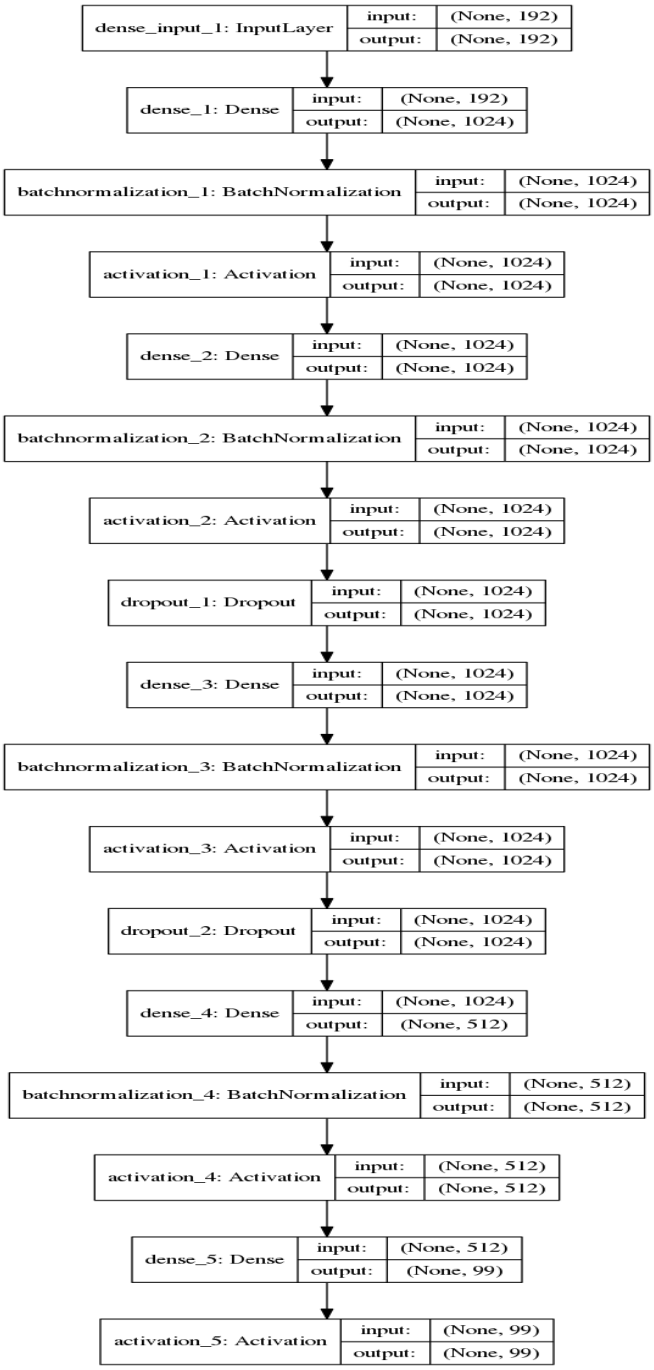


Figura 2

3 Resultados

O modelo final, que é arquitertura descrita anteriormente sendo executada em 130 épocas, com batch size de 128 e todos os outros valores no valor default obteve um resultado final que pode ser analisado nas figuras 3 a 5.

A figura 3 analisa os resultados obtidos pelos folds no cross-validation enquanto nas figuras 4 e 5 mostram os resultados obtidos no modelo final, onde os dados são dividos em 80% para treinamento e 20% para validação.

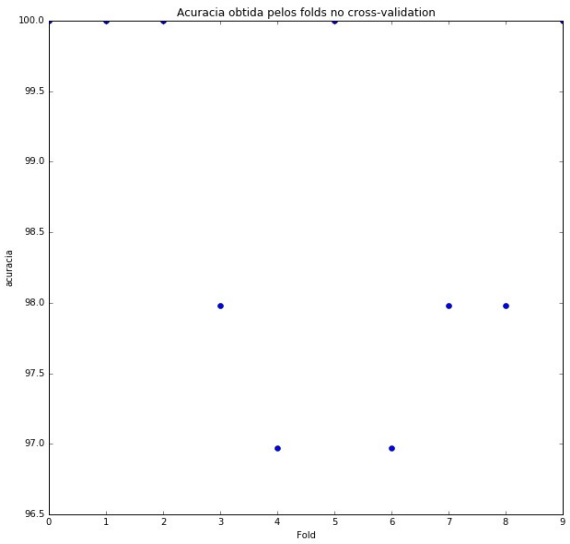


Figura 3

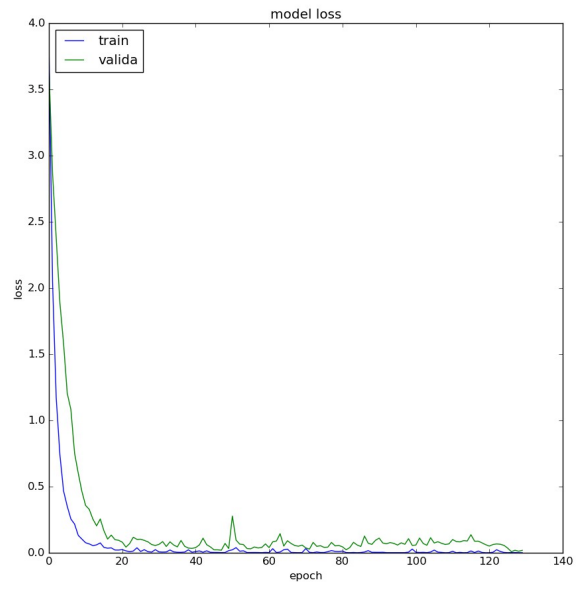


Figura 4

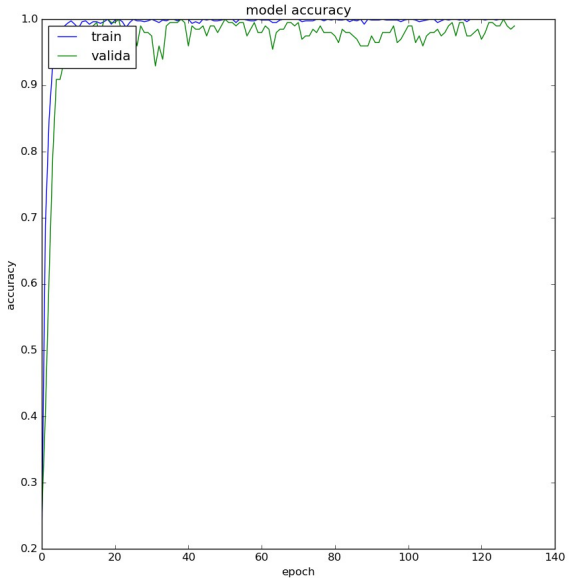
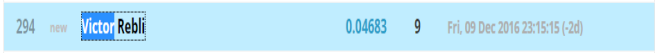


Figura 5

Durante a fase de treinamento, a rede obtêm um bom nível de generalização e consegue bons resultados, chegando próximo de 100% de acurácia, a média obtida no cross-validation foi de 98.78%.

O resultado final obtido no kaggle pode ser verificado na figura 6.



O resultado final obteve a posição final de **0.04683** de multiclass loss, obtendo a 294ª posição.

4 Referências

[1] https://keras.io/
[2] https://www.tensorflow.org/
[3] http://deeplearning.net/software/theano/