Inteligência Artificial
3º ano do Mestrado Integrado em Engenharia Informática e Computação (MIEIC)

E3: Redes Neuronais para a predição de espécies de anuros

Relatório Final

Grupo E3_2

Elementos do Grupo:

António Cunha Seco Fernandes de Almeida - 201505836- up201505836@fe.up.pt
Diogo Luis Rey Torres - 201506428 - up201506428@fe.up.pt
João Paulo Madureira Damas - 201504088 - up201504088@fe.up.pt

20 de maio de 2018

1.Objetivo

Este trabalho tem como objetivo a aplicação de Redes Neuronais para a predição de espécies de anuros, com base nos seus chamamentos.

Para tal foi utilizado um data set com atributos acústicos extraídos dos chamamentos efetuados pelos anuros, em função da sua família, género e espécie.

Os anuros estão diretamente relacionados com o ecossistema onde vivem, sendo a sua atividade avaliada por biólogos como indicador de stress ambiental.

As redes neuronais artificiais são modelos simplificados do sistema nervoso central do ser humano. Trata-se de uma estrutura extremamente interconectada de unidades computacionais, frequentemente designadas por neurônios ou nós, com capacidade de aprendizagem. Durante o processo de aprendizagem, dado por um algoritmo de aprendizagem ou de treino, o peso das conexões é ajustada de forma a se atingir um desejado objetivo.

Deste modo, foi desenvolvida uma rede que consiga identificar espécies de anuros em função de dados pré-processados, com uma elevada percentagem de eficácia.

2.Especificação

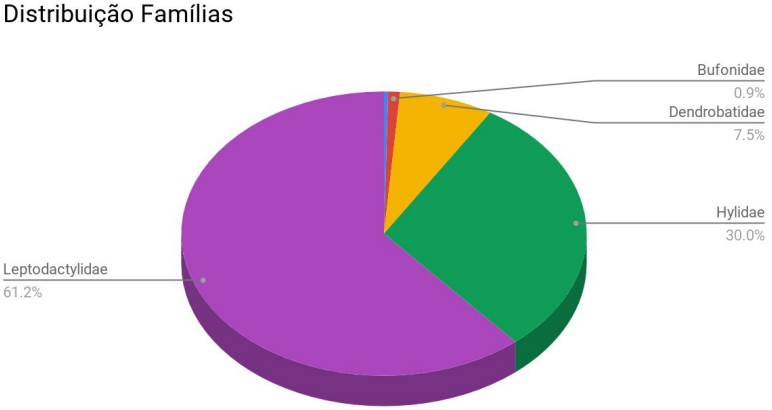
2.1. Descrição e análise do dataset.

O *dataset* utilizado intitula-se “Anuran Calls (MFCCs)”, foi retirado do repositório da UCI (Universidade da California, Irvine) [1], e contém registos de áudio de chamamentos de anuros. Conta com 4 famílias, 8 géneros, e 10 espécies, sendo que cada registo corresponde a um anuro.

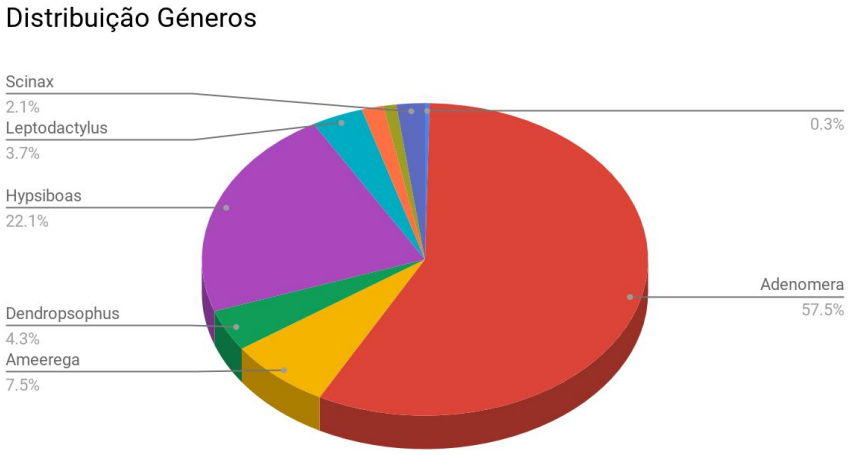
É um *dataset* multilabel, com 3 colunas: família, género e espécie. Contém ainda uma coluna extra com o identificador da gravação. A distribuição das *labels* é a seguinte:

Familías	**Ocorrências**
Bufonidae	68
Dendrobatidae	542
Hylidae	2165
Leptodactylidae	4420

Legenda 1: Famílias presentes no dataset

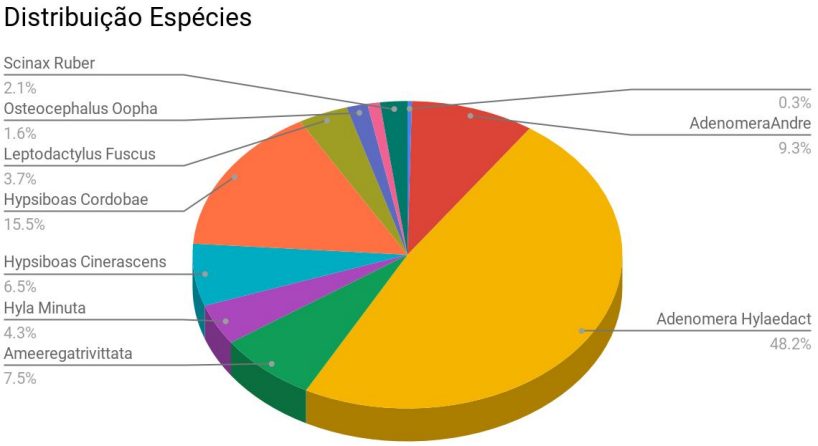


Géneros	Ocorrências
Adenomera	4150
Ameerega	542
Dendropsophus	310
Hypsiboas	1593
Leptodactylus	270
Osteocephalus	114
Rhinella	68
Scinax	148



Legenda 2: Géneros presentes no dataset

Espécies	Ocorrências
AdenomeraAndre	672
Adenomera Hylaedact	3478
Ameeregatrivittata	542
Hyla Minuta	310
Hypsiboas Cinerascens	472
Hypsiboas Cordobae	1121
Leptodactylus Fuscus	270
Osteocephalus Oopha	114
Rhinellagranulosa	68
Scinax Ruber	148



Legenda 3: Espécies presentes no dataset

Analisando a distribuição dos dados, verifica-se uma frequência desequilibrada de dados, sendo 61%, 57%, e 48% dos registos de, respetivamente, Famílias, Géneros, e Espécies serem do mesmo tipo. Isto abre a possibilidade de espécies pouco frequentes na

no *dataset* na fase de testes não serem corretamente identificadas, para além de uma geral tendência em favor dos *labels* de maior ocorrência.

Para a obtenção deste dataset, foi ainda necessário aos autores pré-processar os dados obtidos pelos aparelhos de medição desde a filtragem até a respetiva segmentação em sílabas. Posteriormente foi realizada a sua extração aplicando Mel-frequency Cepstral Coefficients (MFCCs) obtendo assim os valores dos atributos presentes em cada coluna na tabela. Os coeficientes foram normalizados para valores entre -1 e 1.

2.2. Modelo de aprendizagem a aplicar

No desenvolvimento do trabalho foi utilizada uma rede neuronal do tipo **Multilayer Perceptron** (MLP). Este modelo de rede é do tipo *feedforward*, ou seja, a informação flui apenas num sentido: começa nos neurónios de input, passando pelos neurónios intermédios e acaba nos neurónios de output. Os neurónios intermédios formam as camadas intermédias (*hidden layers*). O número de *hidden layers* e o número de neurónios em cada uma é variável, mas cada neurónio está sempre ligado a todos os neurónios da camada seguinte, sendo que a todas as ligações está associado um peso que determina o nível de ativação do neurónio seguinte.

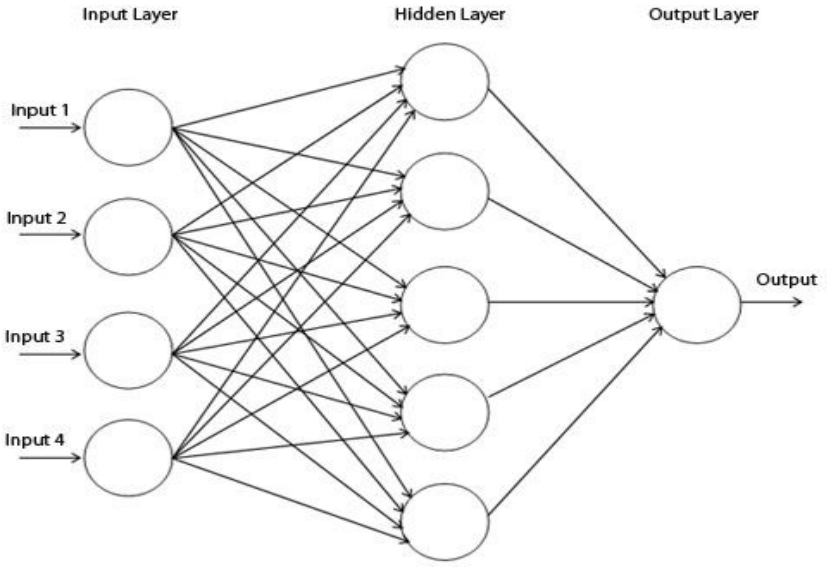
A aprendizagem é supervisionada. Para tal, é utilizado o algoritmo Backpropagation, que consiste em duas etapas de propagação em sentidos inversos: inicialmente, dado um input, o estado é propagado no sentido direto (entrada -> saída) com a ajuda de uma função de transferência. Existem diferentes funções de transferência: tipicamente é utilizada a função sigmóide. No entanto, existem outras possibilidades, como Rectified Linear Unit (ReLU), Tanh (tangente hiperbólica) e Maxout. No âmbito deste projeto foi utilizada a função ReLU. O resultado da propagação produz o output determinado pela rede, que é comparado com o valor real. Em caso de igualdade, não são precisas alterações. Caso contrário, é calculado o erro através da derivada da função de transferência e o valor propagado no sentido inverso, ajustando os pesos das ligações de forma a minimizar a função de perda. De forma a que as variações nos pesos não oscilem demasiado e prejudiquem o processo de aprendizagem em caso de saltos incorretos, no ajuste dos pesos é utilizado um fator de aprendizagem, um valor entre 0 e 1, de forma a atenuar estas variações. O procedimento é repetido até a rede convergir para um estado estável.

2.4. Redes neuronais: arquitectura e configuração da rede

Numa rede *feedforward*, como a que será utilizada, os neurónios estão organizados em camadas. Cada neurónio está ligado exatamente a todos os neurónios da camada seguinte (exceto os de saída). Assim sendo, pode-se dizer que a informação flui num sentido, correspondendo a informação de saída de uma camada à informação de entrada na próxima.

Ao nível das camadas, este tipo de redes são constituídas por **uma input layer**, primeira camada da rede, para onde é passada a informação inicial (assim sendo, o número de neurónios corresponde ao tamanho de dados de input), **uma ou mais hidden layers**, cada uma com um número de neurónios variável, auxiliadoras da deteção de padrões na informação e **uma output layer**, onde as conclusões finais (resultado do reconhecimento)

são guardados. O número de neurónios varia consoante o tipo de resultado que se pretende obter.



Legenda 4: Exemplo de rede feedforward com uma camada de cada tipo.

Embora inicialmente previsto o desenvolvimento de 2 redes - uma com um único *output*, correspondente à label de Espécie, por ser mais específica que as restantes; e uma com 3 *outputs*, correspondentes à totalidade das *labels* - após pesquisa e testes efetuados, conclui-se que a informação adicional não influencia os resultados da identificação de espécies, até porque, para uma dada espécie, o género e a família são constantes, logo a informação é redundante. Contudo, os *labels* extra seriam úteis para obter informação acerca da proximidade acústica entre espécies da mesma família, ou até entre famílias [4], mas isso foge do âmbito deste projecto.

Desta forma, a arquitetura base da rede utilizada foi a seguinte:

1. *Input Layer* com 22 neurónios, correspondentes às colunas de cada registo do *dataset;*
2. *Hidden Layers:*
    a. *Fully Connected Layer (Dense)* com 22 neurónios com ReLU como função de transferência
    b. *Fully Connected Layer (Dense)* com 32 neurónios com ReLU como função de transferência
3. *Output Layer* com 10 neurónios, correspondentes aos 10 diferentes tipos de Espécies, com *softmax* como função de ativação

3.Desenvolvimento

3.1 Ferramentas utilizadas

O trabalho foi desenvolvido em Python utilizando maioritariamente a biblioteca de redes neuronais Keras [2], com *backend* em Tensorflow [3]. Esta combinação foi escolhida pelo foco da mesma em apresentar uma interface simples mas robusta no que toca às redes neuronais, o que permitiu uma atenção maior ao desenvolvimento da arquitetura da rede e métricas associadas, em vez de em detalhes de implementação.

Adicionalmente, foi utilizada a biblioteca scikit-learn [5] em diversas alturas, nomeadamente no pré-processamento dos dados, e no cálculo de métricas adicionais que não foram possíveis através do Keras, como *Precision* e *Recall*. Foram ainda utilizadas as bibliotecas Pandas [6] e Matplotlib [7] para tratamento inicial de dados, e desenho de gráficos, respetivamente.

3.2 Ambiente de desenvolvimento

O trabalho foi desenvolvido recorrendo a um Jupyter [6] *notebook*, para fácil visualização de resultados e separação de código, e é também o formato da entrega do projeto.

3.3. Pré-processamento dos dados.

Inicialmente, por não ter sido identificada utilidade na mesma, foi retirada a coluna do identificador de gravação.

Após análise aos dados do *dataset*, em particular à *label* de Espécies, decidiu-se efetuar *undersampling* à classe maioritária - “Adenomera Hylaedact” - dadas os desequilíbrios descritos na secção 2.1, de forma a evitar *overfitting*. O *undersampling* efetuado reduziu a classe maioritária a 1121 *entries*, reduzindo a ocorrência da mesma de 48% para 18%. Foram também realizados testes com *oversampling* da classe minoritária - “Rhinellagranulosa” - sem grandes resultados, presumivelmente pela dimensão reduzida do *dataset*.

Depois de isoladas colunas correspondentes às *labels*, e dado que são *strings*, foi aplicada uma conversão “One Hot” às mesmas, para poderem ser utilizadas na rede.

Apesar de estar inicialmente previsto uma divisão simples do *dataset* entre dados de treino e dados de teste (com por exemplo, uma divisão 70/30), após uma pesquisa mais aprofundada, decidiu-se implementar *k-fold-cross-validation*, como será descrito na secção seguinte.

4.Experiências

As redes serão treinadas e testadas usando informação proveniente de um único *dataset*. Desta forma, é necessária a divisão do mesmo em conjuntos independentes para

testar a eficácia do modelo, sem risco do mesmo ser tendencioso em relação aos seus dados de treino.

Assim, como introduzido na secção 3.3, foi utilizado *k-fold-cross-validation* em detrimento de uma simples divisão entre dados de treino e dados de teste, mais concretamente *stratified-k-fold*. *K-fold-cross-validation* consiste na divisão igualitária do *dataset* em k *folds*, onde k-1 *folds* são utilizados como dados de treino, e o restante como dados de teste. A rede é então treinada com base nessa divisão. Esto processo é feito k-1 vezes, alterando a combinação de *folds*, e as métricas finais são a média das métricas das redes intermédias. No caso do trabalho, é utilizado k = 5, por ter sido um ponto de equilíbrio satisfatório entre tempo de treino da rede e resultados obtidos. O uso da versão *stratified* prende-se com o desejo de tentar garantir uma representação relativamente igualitária de todas as classes nos diferentes folds, de modo a tentar evitar um *bias* em relação a uma ou mais Espécies.

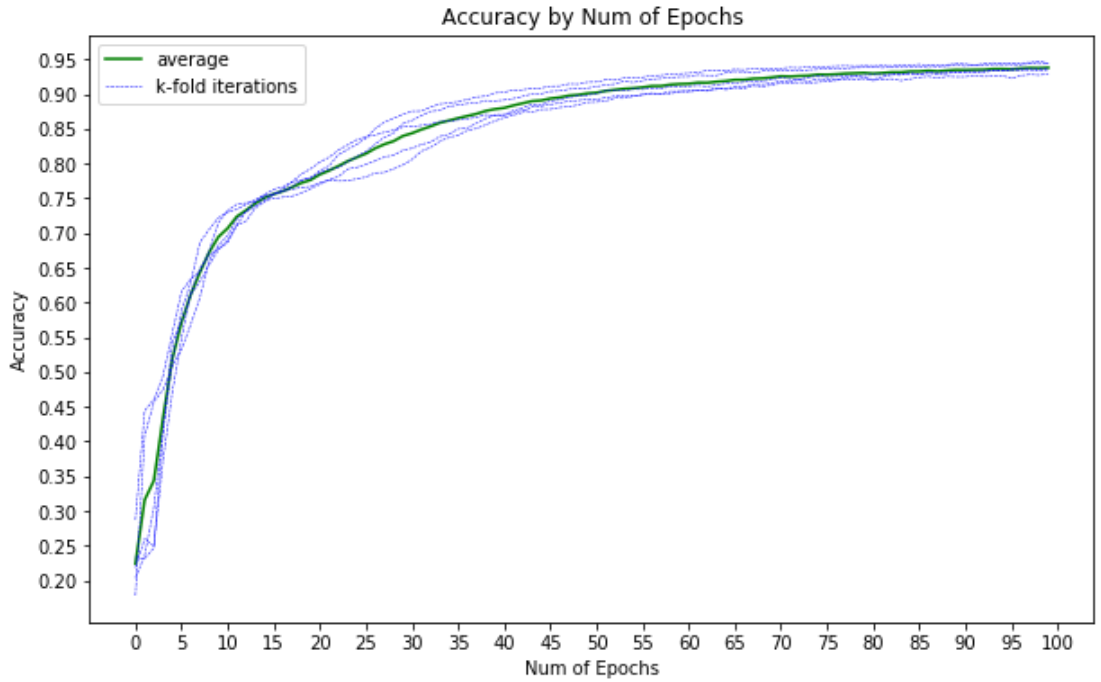
Adicionalmente, durante o processo de treino, o *fold* atual é utilizado como conjunto de validação da iteração atual, de forma a aperfeiçoar as camadas intermédias da rede.

De forma a avaliar a performance do modelo obtido, foram medidas diferentes métricas adequadas para problemas de classificação, como o tratado neste trabalho. Mais especificamente, foram medidos os valores de *accuracy*, *loss*, *recall* e *precision*.

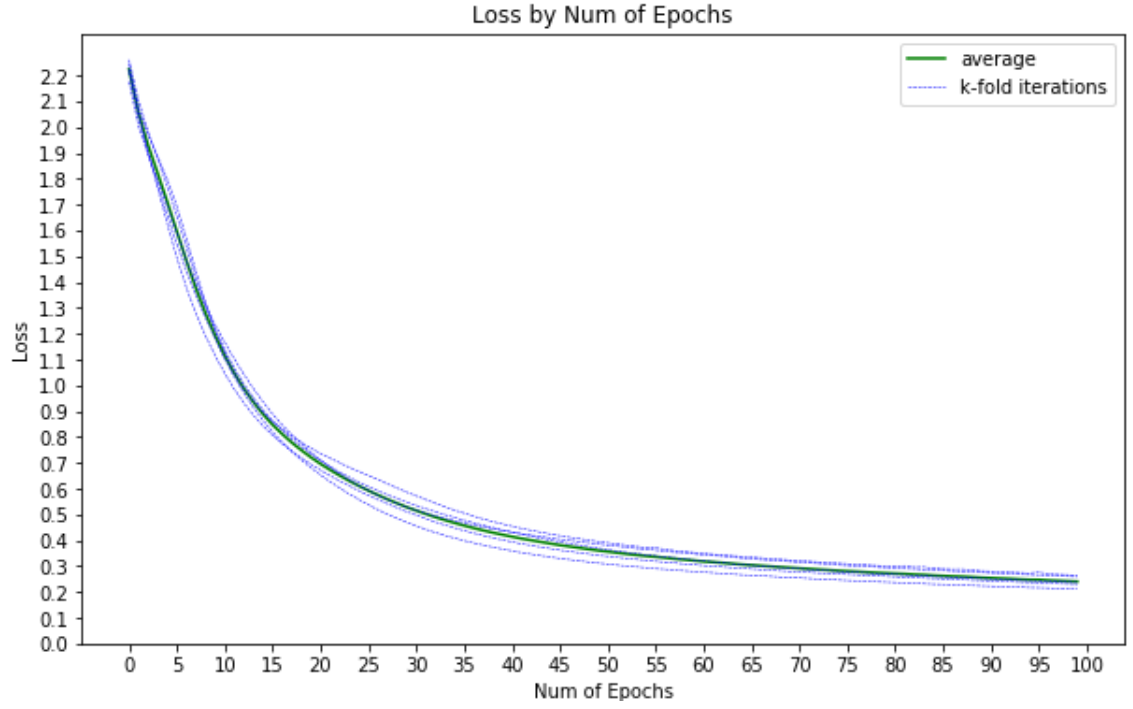
Loss é uma métrica associada ao cálculo do erro durante o ajuste do peso das conexões. Tipicamente diminui ao longo do período de treino, estabilizando o peso das ligações. *Accuracy* mede o rácio de previsões corretas. Não representa uma métrica fiável por si mesma: se o modelo for treinado para classificar sempre da mesma forma, o seu valor será elevado, apesar de não possuir poder preditivo. Assim, usam-se mais duas métricas: *precision*, que mede a percentagem de classificações positivas que, efetivamente, o são, e *recall*, que averigua a percentagem de casos de uma determinada classe que foram classificados como sendo da mesma. Isoladas não têm grande valor, no entanto, combinadas, oferecem uma melhor estimativa da performance do modelo (por exemplo, é fácil atingir o valor 1 para *recall* classificando todos os casos como sendo de uma dada classe, no entanto isso provavelmente causaria um valor bastante baixo de *precision*).

Após testes com várias configurações em redor da arquitetura base da rede, optou-se por correr os testes para dados finais com um *batch size* de 32 e 100 iterações por cada *fold*, por se ter verificado ser um bom ponto de equilíbrio entre tempo de treino e resultados obtidos. Assim, de uma perspectiva geral, com os parâmetros descritos, foi obtido um *accuracy* de 94%, e um *loss* de 0.24.

Representado nas figuras 5 e 6 estão os valores de, respetivamente, *accuracy* e *loss*, em função do número de iterações do processo de treino. Em ambos os casos, cada linha azul representa um *fold*, e a linha verde a média das mesmas. Ao longo das iterações, pode-se verificar a estabilização dos valores de cada *fold* em torno do valor médio, à medida que a função de perda se aproxima de um mínimo local.



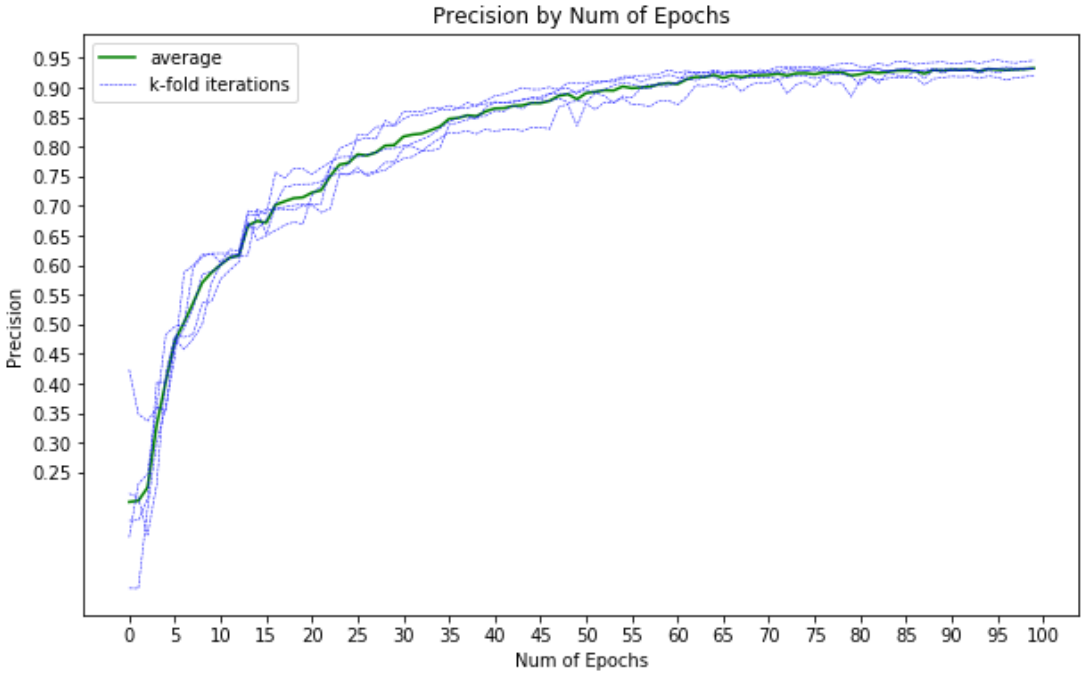
Legenda 5: *accuracy* da rede em função do nº de iterações de treino.



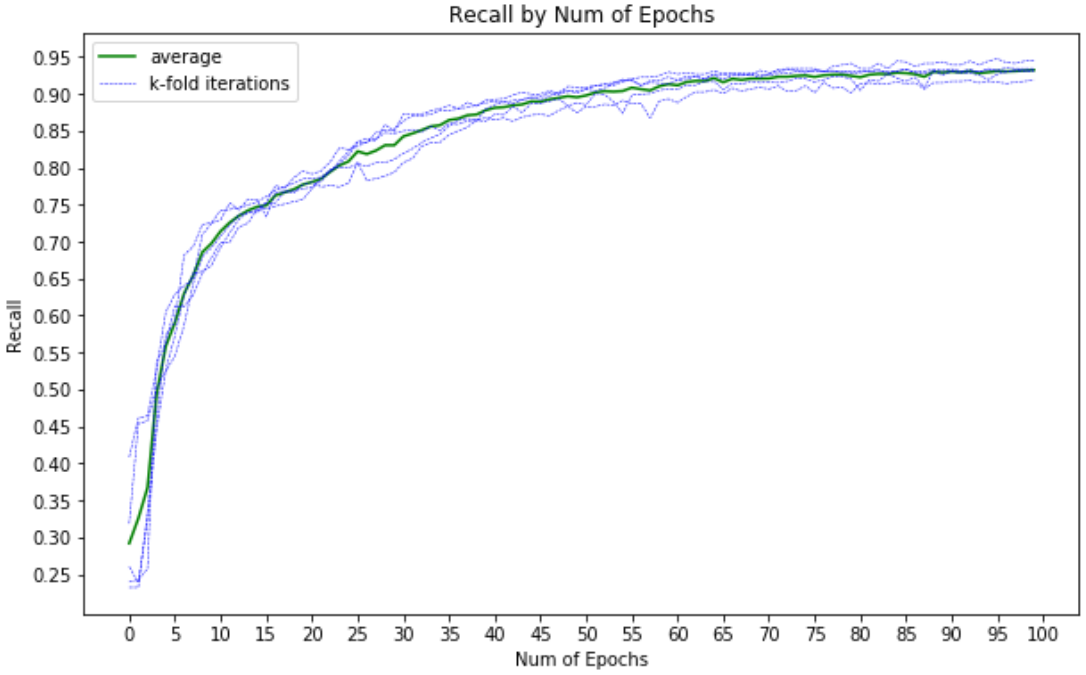
Legenda 6: *loss* da rede em função do nº de iterações de treino.

Para além de, geralmente, *accuracy* por si só não ser uma métrica fiável, no contexto deste problema é acrescidamente relevante analisar outro tipo de métricas, dada a quantidade de classes (Espécies) que constam do *dataset*. Desta forma, o desenvolvimento da arquitetura da rede e especificação de parâmetros de treino tiveram também em conta a otimização da *precision*, *recall*, e consequentemente, *f1-score*. Assim, com a configuração supra descrita, obteve-se uma *precision* de 93% e *recall* de 93%.

Representado nas figuras 7 e 8 estão os valores de, respectivamente, *precision* e *recall*, em função do número de iterações do processo de treino. Como anteriormente, cada linha azul representa um *fold*, e a linha verde a média das mesmas.



Legenda 7: *precision* da rede em função do nº de iterações de treino.

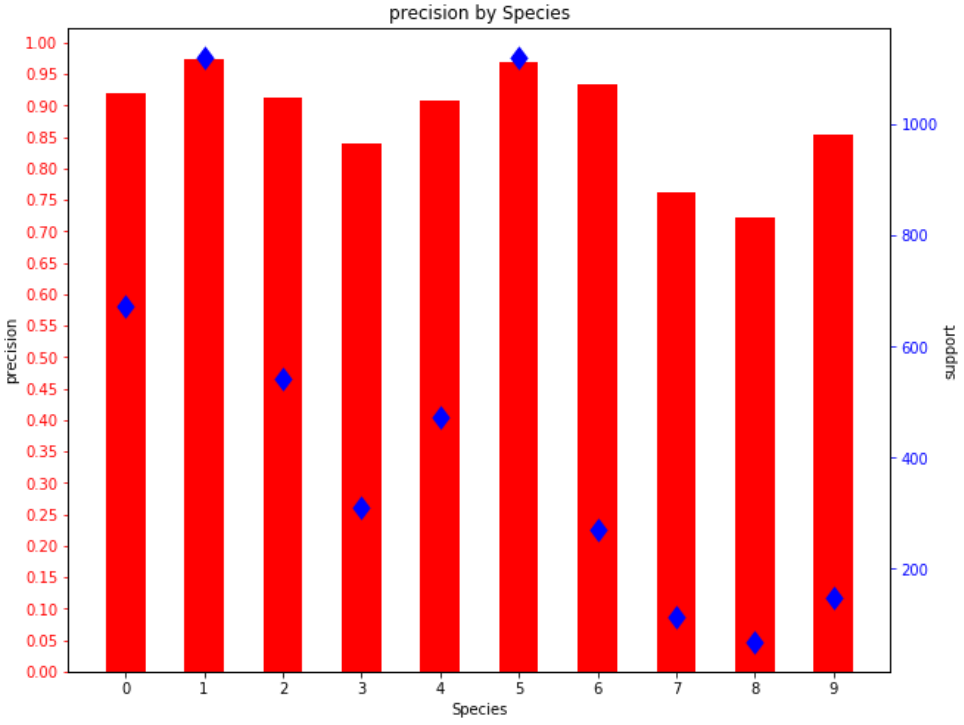


Legenda 8: *recall* da rede em função do nº de iterações de treino.

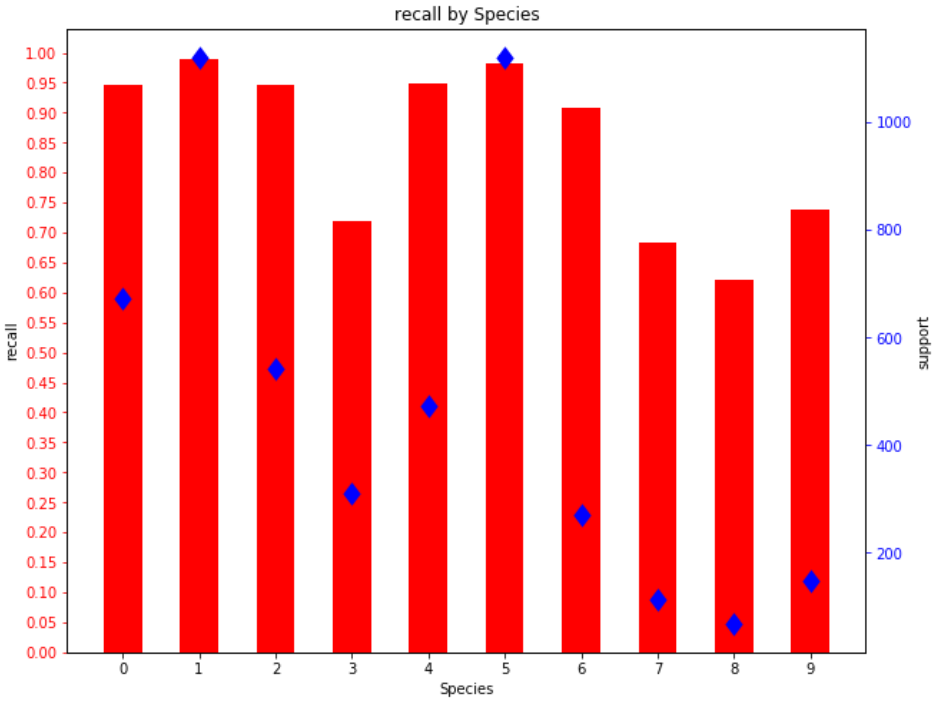
Apesar das métricas obtidas até este ponto provarem-se úteis para tirar conclusões acerca da geral performance da rede, estes não consideram os resultados das previsões por Espécie, individualmente. Assim, foram ainda calculados dados relativos a essa situação, tendo em conta também o número de ocorrências de cada Espécie no *dataset*, para verificar se existe algum tipo de relação entre o número de ocorrências de Espécie, e o valor de métricas relevantes, nomeadamente *precision* e *recall*. Seria expectável que ambas

as métricas fossem diretamente proporcionais ao número de ocorrências, mas isso não se verificou. Na verdade, as classes minoritárias obtiveram valores bastante satisfatórios (acima de 65%), tendo em conta as ocorrências extremamente diminutas das mesmas. Adicionalmente, as classes maioritárias obtiveram valores altos - acima de 90% em ambas as métricas.

Representado nas figuras 9 e 10 estão os valores de, respectivamente, *precision* e *recall*, em função da Espécie. Adicionalmente, o número de ocorrências totais de cada Espécie está assinalada com um *marker* azul, segundo a escalada apresentada à direita.



Legenda 9: *precision* da rede em função do tipo de Espécie



Legenda 10: *recall* da rede em função do tipo de Espécie

Desta forma, verificamos que a rede não só tem uma boa performance geral, mas também consegue obter resultados aceitáveis até em situações para as quais teve poucos dados para conseguir uma generalização sólida.

5.Conclusões

Neste trabalho foi desenvolvida uma rede neuronal capaz de prever espécies de anuros consoante os seus chamamentos. O *dataset* utilizado, apesar de conter uma amostra razoável de casos, era dominado, em quantidade de exemplos, por uma só espécie. Assim, a fim de evitar *overfitting* para a mesma, foi realizado *undersampling*, reduzindo efetivamente a amostra de trabalho.

Em suma, a partir dos resultados das experiências efetuadas ao nível de validação do modelo concebido, é possível concluir que o modelo é satisfatório (valores de *precision* e *recall* acima de 90% e, mesmo considerando as métricas por espécie, os valores são bastante aceitáveis dado o número reduzido de casos em algumas espécies). Apesar de algumas melhorias poderem ser feitas, o resultado final é positivo e a rede obtida permite uma boa classificação de novos casos.

6.Trabalho futuro

Algumas melhorias podem ser efetuadas na conceção do modelo, nomeadamente a fim de aumentar a performance nas classes minoritárias. Um melhor estudo acerca do impacto de diferentes combinações de *undersampling* também seria proveitoso.

7.Recursos

[1] UCI Machine Learning Repository - http://archive.ics.uci.edu/ml/datasets/Anuran+Calls+%28MFCCs%29

[2] Keras - https://keras.io/

[3] Tensorflow - https://www.tensorflow.org/

[4] Colonna, J. G.; Gama, J.; Nakamura, E. F, *Recognizing Family, Genus, and Species of Anuran Using a Hierarchical Classification Approach*. (2016, julho). Draft disponível em https://repositorio.inesctec.pt/bitstream/123456789/5343/1/P-00M-3X5.pdf

[5] scikit-learn - http://scikit-learn.org/stable/

7.1. Percentagem de trabalho de cada elemento

- António Almeida - 1/3
- Diogo Torres - 1/3
- João Damas - 1/3

8.Apêndice

8.1 Manual do utilizador

Para correr o projeto, é necessário ter o projeto Jupyter instalado. Após instalado, correr uma instância do notebook através de *jupyter notebook*, selecionado posteriormente o ficheiro que contém o código fonte a executar.