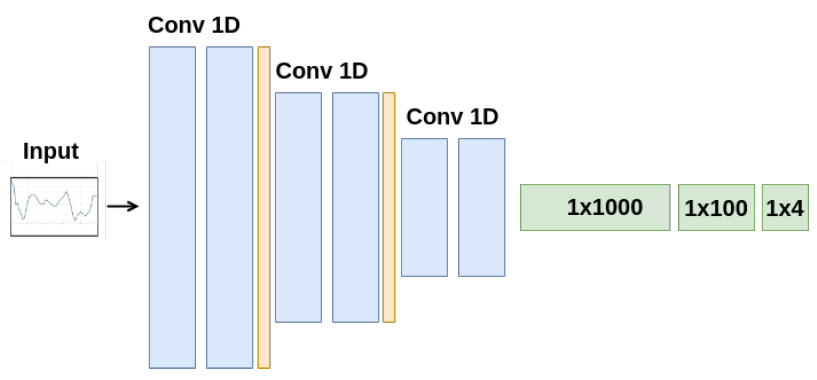
**Deep Convolutional Neural Networks for Shark Behavior Analysis**

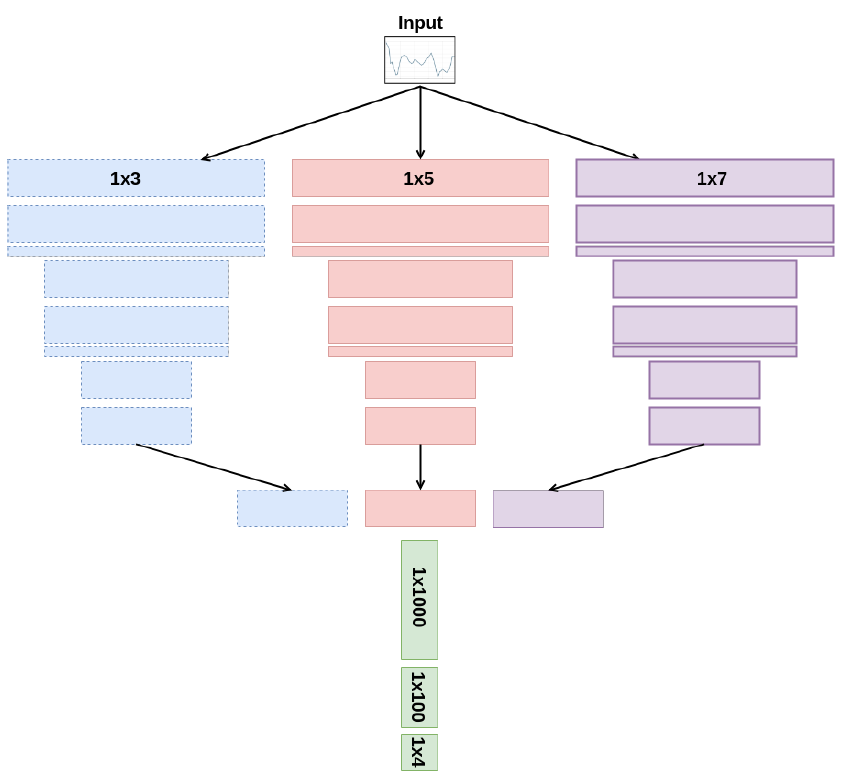
* O estudo do comportamento dos animais implica a sua identificação e estimação das suas diferentes atividades. Com isto é possível, por exemplo, identificar padrões comportamentais
* Este artigo propõe a utilização de modelos de deep learning para classificar automaticamente 4 comportamentos relativos aos tubarões: resting, swimming, feeding e non-deterministic movement (NDM)
* Os dados utilizados para o seu treino são relativos ao sinal da overall dynamic body acceleration (ODBA), que sua vez está diretamente relacionado com os gastos de energia do animal
* Os dados relativos à aceleração (segundo a teoria da relatividade geral) são obtidos utilizando um acelerómetro (ADL – accelerometer data logger), que consegue obter o valor da aceleração em cada uma das dimensões x, y e z. ODBA = soma dos valores absolutos em cada uma das dimensões
* Utilizando dados exemplo do valor da ODBA em períodos pequenos de tempo, foram treinadas e avaliadas três CNN’s com arquiteturas diferentes com o intuito de classificar o comportamento nesse período (2s neste caso) -> classificação de time series extraindo características temporais automaticamente

**Arquitetura das Redes Propostas**

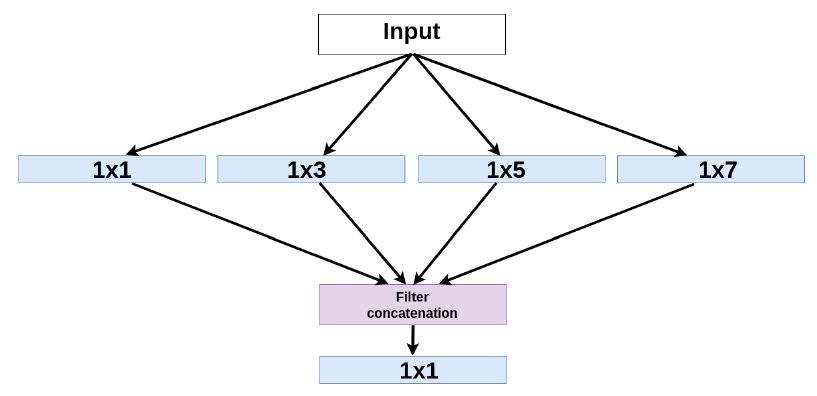
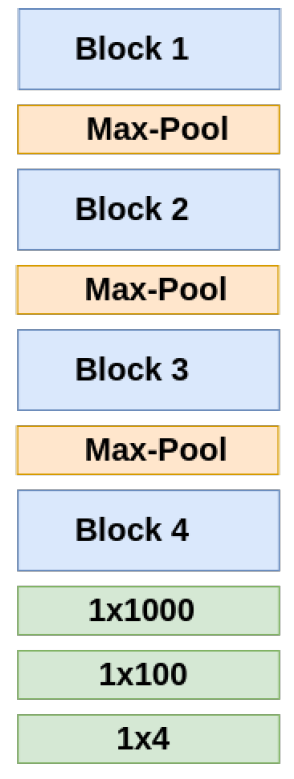
* A arquitetura das redes desenhadas é baseada na rede VGGNet, que por sua vez é uma extensão da Alexnet. A VGGNet é mais profunda e utiliza kernels mais pequenos, ou seja, é uma rede mais profunda mas com menos parâmetros para se treinar
* Primeira rede: SharkVGG1 -> 9 camadas com kernels 5x1
  + 2 camadas de convolução seguidas de uma camada de max pooling (as seguintes camadas de convolução aplicam o dobro dos filtros das anteriores)
  + A função de ativação utilizada foi a ReLu e é aplicada também batch normalization
  + Após as camadas de convolução, seguem-se 2 fully connected layers (1000 e 100 neurónios respetivamente) e a camada de output (4 neurónios)



* Segunda rede: SharkVGG 2 -> as camadas relativas à extração de características do primeiro modelo são aplicadas com diferentes tamanhos de kernels, 3 5 e 7 e o output da última convolution de cada uma é concatenado, antes de ser passado à primeira hidden layer

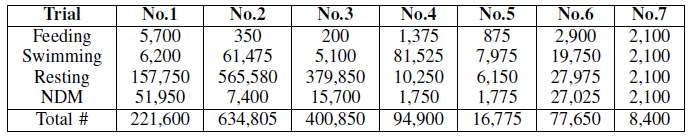


* Terceira rede: SharkInception -> utiliza o conceito de inception module. Em cada bloco de convolução: são aplicados sobre o input diferentes filtros com diferentes kernels sizes cujos outputs são agregados e passados para a próxima layer. Isto tem a vantagem de ser possível processar informação na time serie em diferentes níveis de detalhe em paralelo



**Resultados e Avaliação**

* O dataset é composto por dados recolhidos em 7 trials de 2h cada. Devido ao próprio comportamento do peixe, é normal que as amostras relativas a cada atividade não estejam balanceadas. Para ser efeito são aplicados métodos de data augmentation
* Os peixes relativos a cada amostra devem estar uniformemente distribuídos pelos diferentes conjuntos (treino, validação e teste)
* Neste trabalho, foram utilizadas as amostras relativas a 5 trials como treino, 1 para validação e 1 para teste
* Número de amostras:



* Também foi treinado uma Random Forest utilizando estatísticas de cada time serie, para efeitos de comparação com os modelos de deep learning
* Métricas: Accuracy, Precision, Recall e F1-score
* O melhor resultado foi obtido na primeira rede com 76% accuracy, sendo que os valores demonstram que todos os modelos entraram em overfitting

**Alguns Conceitos**

* Batch: conjunto de samples a serem passados pela rede até haver uma atualização dos pesos. Se for muito alto o cálculo do gradiente vai ser mais accurate mas levará mais tempo a treinar. A abordagem estocástica coloca batch size=1, sendo os pesos atualizados a cada sample
* Batch Normalization: se houver um dos pesos, de um determinado neurónio, que seja excessivamente elevado o seu output por sua vez também será elevado e isto será propagado pelas restantes camadas da rede, causando instabilidade no treino e pior performance. Uma das soluções é normalizar os valores de output de uma determinada camada.