# REDES NEURAIS UTILIZANDO TENSORFLOW E KERAS

Como facilitar o desenvolvimento de redes neurais com frameworks

RIBEIRO, Maxwell M. [1]
GUIMARÃES,Samuel S.[1]
[1] Acadêmico de Ciência da Computação, UNIFENAS

**RESUMO**.

Projeto de Inteligência Artificial tomando base o artigo realizado por Ramos (2016), que a rede neural criada por ele permite reconhecer a autenticidade de condutores autorizado, com isso projetamos uma rede neural com base em deep learning, usando frameworks TensorFlow e a API Keras que o intuito deles é levar para jovens programadores conseguirem criar sua própria rede neural sem com maior facilidade, usando abstração de cálculos que são necessário em uma deep learning como hoje em dia, com base nas bibliotecas TensorFlow e Keras na Linguagem de programação Python, que facilita ainda mais no desenvolvimento de softwares que podem tomar suas próprias decisões, uma das maiores dificuldades do desenvolvimento de Inteligência Artificial é entender como podemos simular o comportamento humano passando para a máquina, neste artigo mostraremos que é possível criar uma rede neural artificial, com estes frameworks facilitadores no desenvolvimentos dos projetos.

**ABSTRACT**

Artificial Intelligence Project based on the article by Ramos (2016), which the neural network created by him allows to recognize the authenticity of authorized drivers, with that we designed a neural network based on deep learning, using frameworks TensorFlow and API Keras that their aim is to make it possible for young programmers to create their own neural network without more easily, using the abstraction of calculations that are required in a deep learning as it is today, based on the TensorFlow and Keras libraries in the Python programming language, which facilitates even more in the development of software that can make their own decisions, one of the major difficulties of the development of Artificial Intelligence is to understand how we can simulate human behavior passing to the machine, in this article we will show that it is possible to create an artificial neural network with these frameworks facilitators in the development of projects.

**PALAVRAS-CHAVE**

Redes Neurais Artificiais, Deep Learning, Machine Learning. classificação.

**KEYWORDS**

Artificial Neural Networks, Deep Learning, Machine Learning, classification.

## 1 Introdução

Neste artigo serão abordados conteúdos da Inteligência Artificial (IA), que é uma tecnologia que está crescendo exponencialmente a cada dia, sendo utilizadas em várias empresas grandes como a google e a Microsoft.

Quando se fala em IA, muitos já associam com robôs autônomos que irão substituir o ser humano aqui na Terra, porém, a realidade que muitos não percebem é que a Inteligência se apresenta não só na automação, mas ele está presente em nosso dia-dia, como exemplo, quando se acessa a Netflix e na tela principal é recomendado vários filmes que te agrada. Para isso é feita uma análise e um processamento inteligente com base no seu histórico de visualizações que procura encontrar um padrão de gênero de Filmes.

Outro exemplo é o corretor do seu celular que vai completando as palavras conforme está sendo digitada. Isso só é possível por conta de IA, que trabalha em um reconhecimento de padrão e indica a palavra possível a ser escrita, antes mesmo que seja completada pelo usuário.

Esta tecnologia pode ser definida como a capacidade de um software de raciocinar, decidir e tomar decisões. Este termo Inteligência Artificial só se popularizou hoje por conta dos avanços tecnológicos de algoritmos e hardware, porém já existe destes 1956, após a Segunda Guerra Mundial em uma conferência no campus do Dartmouth College, onde surgiu os primeiros conceitos de IA que naquele momento não podiam ser colocados em práticas por contas das limitações da época.

Após este período houve vários avanços. No ano seguinte surgiu o perceptron que é uma rede de uma camada, depois veio Marching Learning com a capacidade da máquina aprender executar tarefas automaticamente e após 80 foi proposto por edward feigenbaum os primeiros sistemas especialista que realizam atividades que são complexas e específicas em um campo, fazendo o papel do humano, só que mais veloz. Com isso a ia se aproxima dos mercados corporativos, despertando os olhares das empresas por sistema inteligentes por suas vastas aplicabilidade.

Depois desses acontecimentos, a IA só foi crescendo cada vez mais, com exemplo, a máquina que derrotou o homem em um jogo de xadrez em 97 ou o reconhecimento de voz pela Google em 2008.

Com este artigo, será apresentado deep learning (Aprendizagem Profundo) de forma mais prática e usual, graças ao frameworks do TensorFlow e a API Keras que nos ajudam e facilitam na criação de uma rede neural profunda sem precisar aprofundar nos cálculos e nos algoritmos complexos. Para quem é novo nessa área, será um incentivo à pesquisa para aprofundar nesta ampla área de Inteligência Artificial.

## 2 Machine Learning e Deep Learning

Para compreender melhor os avanços realizados pela Inteligência artificial, pode-se fazer um comparativo de estilos e objetivo de codificação. Na programação tradicional é desenvolvidas funções com ações bem definidas, como exemplo, guardar as característica de um copo e verificar por um método se a entrada corresponde aquele específico copo. Como podemos vê, é algo bem definido, onde o resultado final é

somente dois: corresponde ou não aquele copo. Agora imagine como seria feita a programação para saber se as características de entradas correspondem a o 'tipo copo', sendo que existem milhares de milhares de copos de características diversificadas. Seria algo extremamente maçante e inviável fazer em uma metodologia tradicional e uma solução para isso é o Machine Learning (aprendizagem de máquina).

A ideia é que computadores seja capaz de fazer suas próprias funções, abstraindo padrões de dados reais que servem para predição ou classificações. Levando em paralelo o exemplo falado acima, com base a uma grande quantidade de dados de copos diversificados, o programa, por meio de funções matemáticas e algoritmos complexos, procura encontrar padrões que gerem um modelo capaz de intensificar copos, tanto copos que ele usou para aprender, quando copos que nunca passaram em sua fase de treinamento.

Existem vários tipos e aplicações com aprendizagem de máquinas. Nesse artigo vamos focar em redes neurais profunda (Deep Learning), usando redes neurais com aprendizagem supervisionados.

Deep Learning tem uma complexidade, porém sua implementação de modelos é muito menos assustadora e difícil por conta dos frameworks disponíveis e é nessa parte que entra o TensorFlow e a API do Keras.

## 3 TensorFlow e Keras

TensorFlow é uma bibliotecas de código aberto criadas pela equipe do Google Brain para aprendizado de máquina e pesquisa de redes neurais profundas. Ele vem crescendo muito pelo fato de seu desempenho por utilizar XLA, um poderoso compilador de álgebra linear que torna a execução mais rápida, rodando em CPUs, GPUs, TPUs e outros.

Muitas empresas grande utilizam essa tecnologia, como a Intel, Uber, DropBox e claro, a própria Google, que além de utilizar, buscar incentivar o uso de seu framework. Com ele pode se aplicar em várias situações, como dois exemplos bem conhecido que é no reconhecimento de imagens e na incorporação de palavras.

Seu maior benefício é sua abstração. Em vez do desenvolvedor lidar com detalhes básicos da implementação do algoritmo, pode-se concentrar na lógica geral da aplicação. Outro ponto positivo do TensorFlow, é que ele é flexível, existem APIs que ajudam ainda mais para o usuário utilizar suas tecnologias e nesse artigo, usaremos uma delas, que é a API Keras.

Keras tem como vantagem ter uma interface mais simples e otimizada para casos mais comuns e ela facilita ainda mais na criação de um modelo de redes neurais profundas, sendo mais fácil de entender.

O seu objetivo é que no futuro a aprendizagem profunda seja uma ferramenta para todos e não somente para pesquisadores e especialista.

Neste artigo iremos utilizar o Keras para criar um modelo capaz de classificar se o motorista é autorizado ou não a dirigir o veículo.

## 4 Metodologia

Neste Tópico, iremos abranger como utilizamos o treinamento da rede, o que foi necessário para o desenvolvimento, como funciona o algoritmo empregado para conseguirmos realizar o aprendizado da nossa rede neural e aplicamos os frameworks

## 4.1 Base de Dados

Para que seja possível alcançar o objetivo de treinar uma rede neural que seja capaz de classificar se o condutor veicular é autorizado ou não em dirigir o automóvel é muito importante ter uma boa base de dados, e a informações utilizadas nesta aplicação foi retirada do artigo Identificador de condutor de veículo utilizando Redes Neurais Artificiais e OBDII, que optou em utilizar um dispositivo OBD-II que é conectado ao computador de bordo do veículo e por meio de uma conexão Bluetooth, transmite dados referente a situação daquele veículo aquele naquele momento.

Para receber esses dados do OBD-II foi utilizado um aplicativo que se chama Torque. Ele gera um arquivo .csv completo com todas informações e para se treinar a rede, foi filtrados tais dados: Posição do pedal do acelerador, aceleração x (reta), aceleração y (subida) e aceleração z (curva). Essas informações também foram tratadas, buscando garantir maior confiabilidade, para que seja possível criar um modelo generalizado.

Foram utilizados ao todo 4297 informações, que foram divididos em dois, 90% para treinamento (3437) e 10% para teste (860).

## 4.2 Implementação da Rede Neural com as bibliotecas

A aplicação foi realizadas na linguagem python e a seguir será mostrado o passo a passo da estrutura, do treinamento e dos testes realizados no modelo gerado.

```
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense
from matplotlib import pyplot
```

Primeiramente é preciso fazer a importação da biblioteca pendas, keras e da classe Sequential, que é para criação da estrutura da rede, que tem como característica de ser ordenada e sequencial (entrada, camada(s) ocultas e saída). Importamos de keras.layers a classe Dense, que representa que cada um dos neurônios é ligado com todos os outros da camada subsequente.

O Pyplot será utilizado para mostrar o gráfico.

```
base = pd.read_csv('entradas.csv')
```

Utilizando a função read_csv do pandas, foi criada uma variável que armazena todos os dados salvos em um arquivo csv.

```
#Separando os previsores e as classes
previsores = base.iloc[:, 0:4].values
classe = base.iloc[:, 4].values
```

Após importar os dados eles foram divididos em previsões ( posição do pedal do acelerador, aceleração x , aceleração y e aceleração z) e classe (resultado se é autorizado ou não). Isto é necessário para o momento que for executar o treinamento dos dados.

```
from sklearn.model_selection import train_test_split
previsores_treinamento, previsores_teste, classe_treinamento, classe_teste = train_test_split(previsores, classe,
test_size=0.2) #0.15
```

Neste exemplo, serão divididos os dados para treinamento e para teste, pois ao terminar o processo de aprendizagem, é gerado um modelo com os pesos ajustados, e sobre eles é aplicado esses dados teste para saber sua eficiência.

```
classificador = Sequential() #Criando uma nova rede neural
#primeira camada oculta
#units: quantidade de neuronios
#activation: função de ativação relu: tem uma detencia a dar resultados melhores
#kernel_initializer: como será a inicialização dos pesos
#input_dim: quantidade de elementos na camada de entrada
classificador.add(Dense(units = 8, activation = 'relu', kernel_initializer = 'random_uniform', input_dim = 4))
```

Neste momento começa a criar a rede neural, para iniciar é necessário uma variável receber Sequential() e depois pode-se começar a definir as camadas. Para isso, acima mostra a criação de uma camada de entrada com 4 neurônios (input_dim, adicionada junto com a primeira camada oculta) e uma camada oculta densa com 8 neurônios com a função de ativação relu e inicialização de pesos uniformemente aleatórios.

Esta função relu é a função de ativação mais utilizadas e ele tem com característica de retornar 0 para os valores negativos e para os positivos eles mesmo.

```
classificador.add(Dense(units = 10, activation = 'relu', kernel_initializer = 'random_uniform'))
classificador.add(Dense(units = 8, activation = 'relu', kernel_initializer = 'random_uniform'))
```

Foram adicionadas mais duas camadas ocultas, seguindo o modelo da anterior, com a diferença que uma foi criada com 10 neurônios.

*classificador.add(Dense(units = 1, activation = 'sigmoid'))*

Esta será a camada de saída, que terá um neurônio e se utilizar a função sigmoid que retornam valores entre 0 e 1, sendo que o mais próximo de zero significa que o condutor veicular não é autorizado e o mais perto de 1 para autorizado.

*classificador.compile(optimizer = 'nadam', loss = 'binary_crossentropy', metrics = ['acc'])*

Para compilar as redes, foram passando alguns parâmetros importante.

Optimezer que é o parâmetro da função de ajuste dos pesos e neste caso o nadam foi o que deu um resultado melhor.

Loss é a função de erro, e como estamos trabalhando com uma classificação binária utilizamos o binary_crossentropy que é um padrão bastante usando.

Metrics é usado para fazer a validação onde é realizado um cálculo para saber a precisão da rede, com base nas classificações corretas e erradas.

*history = classificador.fit(previsores_treinamento, classe_treinamento, batch_size = 10, epochs = 3000)*

Neste momento é feito a classificação da rede usando o método fit que foi passado como parâmetro os dados de treinamento, o batch_size como 10, que é em quantos dados e se ajustar os pesos e a quantidade de épocas que neste exemplo foram utilizadas 3000.

*previsoes = classificador.predict(previsores_teste)*
*previsoes = (previsoes > 0.5)*
*from sklearn.metrics import confusion_matrix, accuracy_score*
*precisao = accuracy_score(classe_teste, previsoes)*
*matriz = confusion_matrix(classe_teste, previsoes)*
*resultado = classificador.evaluate(previsores_teste, classe_teste)*

### 4.3 Testes para avaliação

Para avaliar o modelo criado, foi testar os dados de previsões testes, usando o predict e do resultado informação pela rede, foi convertido os dados para true e false para ser possível comparar com a classe teste. Para ser feito a comparação, foi utilizado a confusion_matriz, que é um método que retorna a quantidade de acertos e erros da rede (acertos: quanto era 1, acertou com 1 - quando era 0 e acertou com 0. erros: quando era 1, errou resultando em 0 - quanto era 0, errou resultado em 2).

## 5 Resultados e Discussão

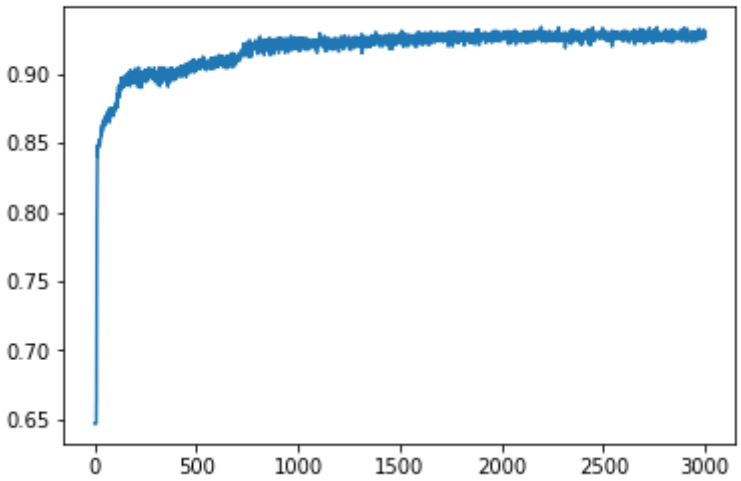


**FIGURA 1 - Acc X Epoch**

Quando se está treinando, a cada época é feita uma avaliação da rede, mostrando se ela está conseguindo ou não aprender. Na FIGURA 1, pode-se observar que teve uma aprendizagem rápida até às 500 épocas e depois foi aprendendo mas devagar, pois quando mais próxima de 1, mais será exigido da rede para melhorar.

Na época 3000 a rede teve um aproveitamento de 0.9264 (92,64%) de acertos, que é consideravelmente bom, só que para ter uma melhor avaliação, é importante fazer outros testes, pois este aproveitamento foi calculado com base nos dados de treinamento, que são os dados que a rede conhece e utilizou para criar o modelo. Agora para ter uma melhor confiabilidade, é preciso testar o modelo gerado com dados novos, que nunca viu para saber se foi criada um modelo genérico, pois pode acontecer que a rede decore e sirva somente para aqueles dados utilizados para treinamento.

| | 0 | 1 |
|---|---|---|
| 0 | 234 | 46 |
| 1 | 22 | 558 |

**FIGURA 2 - Matriz de Confusão**

Para testar a rede com os dados de teste, foi utilizado uma função que gera uma Matriz de Confusão, conforme ilustrado na FIGURA 2. Esta função faz o comparativo do vetor de resultado gerado pela rede e o vetor de resultado real, gerando uma matriz que mostrar os acertos e erros. Neste exemplo a rede acertou 234 vezes onde o condutor não era autorizado, 558 onde o condutor era autorizado e errou 22 vezes onde o veículo era autorizado e a rede falou que ele não era, 46 vezes em que não era autorizado e a rede falou que era autorizado. Dividindo a quantidade de acertos pela quantidade total, a rede chegou em uma precisão de 0.9209 (92,09%), um resultado bem parecido com o que deu na base de dados de teste. Com isso temos mais segurança no desempenho da rede, simulando ela em uma situação que acontecerá quando estiver em produção, que nesse caso, no momento atual que um condutor esteja andando no veículo.

## 6 Conclusão

O uso do Tensorflow e Keras foi muito viável, pois com pouco tempo de treinamento e uma base de dados relativamente baixa, ele teve um resultado de cerca de 92% que nos mostra uma confiabilidade bem alta para tomar decisões, com esse desenvolvimento comprovamos que é possível realizar um rede neural confiável não necessitando de um alto conhecimento no assunto, com ajuda de bibliotecas prontas para facilitar e possivelmente um dia chegarmos ao momento que qualquer pessoa consiga criar sua própria rede neural, mesmo estando iniciando na área de desenvolvimento de softwares inteligentes.

## 7 Referências


AI Google. Keras. 2018. Disponível em: <https://www.tensorflow.org/guide/keras>. Acesso em: 15 jun. 2017.

ALVES, Wellington. Entenda a importância da Inteligência Artificial. 2018. Disponível em: <https://canaltech.com.br/inteligencia-artificial/entenda-a-importancia-da-inteligencia-artificial-100442/>. Acesso em: 05 jul. 2017.

DATASCIENCEACADEMY. O QUE É O TENSORFLOW MACHINE INTELLIGENCE PLATFORM?. 2018. Disponível em: <http://datascienceacademy.com.br/blog/o-que-e-o-tensorflow-machine-intelligence-platform/>. Acesso em: 14 jun. 2017.

ELIEZER, ELIEZER BOURCHARDT. Inteligência Artificial—Um pouco da história e avanços atuais. 2018. Disponível em: <https://medium.com/@eliezerfb/intelig%C3%AAncia-artificial-499fc2c4aa79>. Acesso em: 05 jul. 2017.

GRANATYR, Jones. Deep Learning com Python de A à Z - O Curso Completo. Disponível em: <https://www.udemy.com/deep-learning-com-python-az-curso-completo>. Acesso em: 10 jul. 2017.

HD Store. O que é o TensorFlow? A biblioteca de machine learning explicada. 2018. Disponível em: <https://blog.hdstore.com.br/o-que-e-o-tensorflow/>. Acesso em: 12 jun. 2017.

JOVIAN LIN, Ph.D. Categorical_crossentropy VS. sparse_categorical_crossentropy. 2018. Disponível em: <https://jovianlin.io/cat-crossentropy-vs-sparse-cat-crossentropy/>. Acesso em: 21 jun. 2017.

KERAS. Documentação Keras. 2018. Disponível em: <https://keras.io/why-use-keras/>. Acesso em: 21 jun. 2017.

RAMOS, Celso de Ávila; LACERDA, Wilian Soares; SCALCO NETO, Heitor. Identificador de condutor de veículo utilizando Redes Neurais Artificiais e OBDII. **XIII Brazilian Symposium of Information Systems**. Lavras, 2017. P. 158-65.

SAS. Inteligência Artificial O que é e qual sua importância?. 2018. Disponível em: <https://www.sas.com/pt_br/insights/analytics/inteligencia-artificial.html>. Acesso em: 28 jun. 2017.

SAS. Big Data O que é e qual sua importância?. 2018. Disponível em: <https://www.sas.com/pt_br/insights/big-data/what-is-big-data.html>. Acesso em: 28 jun. 2017.