Aprendizado de máquina em sistemas embarcados

TensorFlow Lite em microcontroladores

Prof. Anderson W. Spengler
anderson.spengler@ufsc.br

About Anderson W. Spengler

- Graduated in Applied Physics at IFGW/UNICAMP
- Masters and Doctors Degree in Electrical Engineering at FEEC/UNICAMP
  - Precision Electronic Instrumentation
  - Modulation/Demodulation of Interferometric Fiber Optic Gyroscope
- Since 2013 - Professor at UFSC/Joinville
  - Mechatronics and Aerospace Engineer
- Three main research topics
  - Electronic instrumentation with Embedded Systems
  - Sensor data fusion for Inertial Navigation Systems
  - Energy Harvesting for Embedded Systems

Evolução dos Sistemas Embarcados

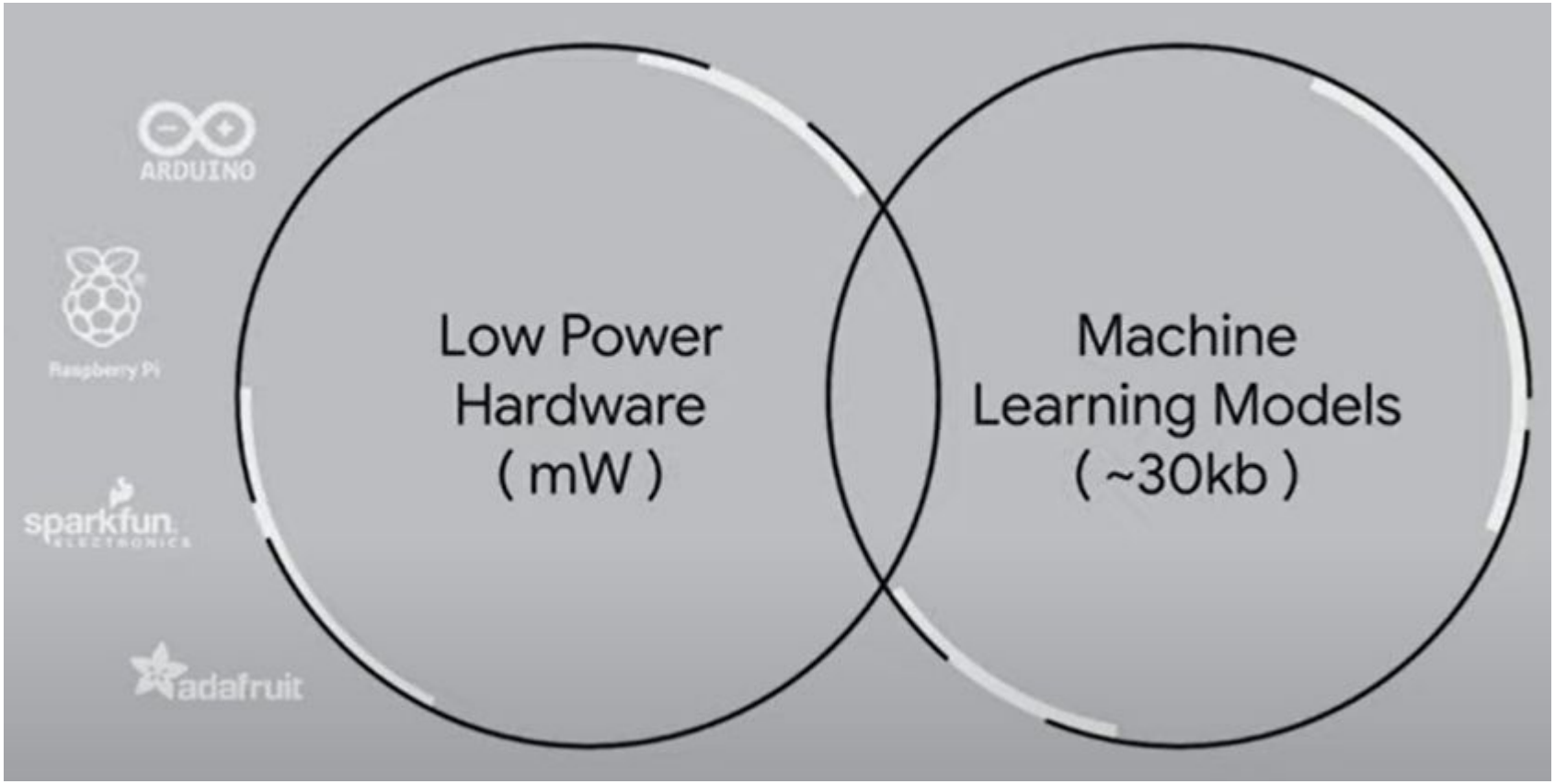
- Uso de sistemas embarcados
  - Conceitos de sistemas ciber-físicos
- Evolução da capacidade de processamento, miniaturização dos sistemas, preço e consumo de energia.
- Pentium 100 - 100 MHz - Março de 1994 - 32 bits - Pico de 10W
- T4MC123GH6PM - 80 MHz - 2014 - ARM Cortex M4F 32 bits - 0,15W com todos os periféricos - 32 kB RAM e 256kB de Flash

TinyML

Tiny machine learning is broadly defined as a fast growing field of **machine learning** technologies and applications including hardware, algorithms and software capable of performing **on-device sensor** data analytics at **extremely low power**, typically in the mW range and below, and hence enabling a variety of always-on use-cases and targeting battery operated devices.

TinyML.org

TinyML + Microcontroladores



Conteúdo dessa aula

- Motivação da criação e uso de algoritmos de ML em sistemas embarcados, principalmente em microcontroladores.
- Conceitos básicos de Machine Learning para utilização de TensorFlow Lite
- Tool-chain para utilização de TensorFlow Lite
- Exemplos de uso

Conceitos Básicos de Machine Learning

- TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers by Pete Warden and Daniel Situnayake
- Introdução ao básico necessário para compreender o fluxo de programação de uma aplicação usando TensorFlow.

Um Exemplo de Problema com ML

- Máquina industrial, às vezes ela quebra e é cara de reparar. Possivelmente adquirindo dados da máquina durante a operação, você pode ser capaz de prever quando ele irá estragar e parar a operação antes de danos. Pode-se adquirir dados como taxa de produção, temperatura e quanto está vibrando. Alguma combinação desses fatores pode indicar o problema. Mas como descobrir qual?

Conceitos Básicos de Machine Learning

- Aprendizado de máquina utiliza dados passados para fazer prever condições.
- Criar um “programa de ML” é diferente do usual
  - Não é feita uma programação processual das saídas em função das entradas.
  - Em ML repassa-se dados a um algoritmo especial e este algoritmo que descobre as regras.
  - O algoritmo cria um modelo do sistema que realiza previsões/inferências.

Conceitos Básicos de Machine Learning

- Várias formas de abordar.
  - Deep learning
    - rede de neurônios simulados (array de números) é treinada para expressar relação entre entradas e saídas.
    - Diferentes arquiteturas, diferentes resultados.
      - Complexidade de análise de imagens e sequência numérica.

Deep Learning Workflow

1. Definir um objetivo
2. Coletar um conjunto de dados
3. Projetar uma arquitetura do modelo
4. Treinar um modelo
5. Converter o modelo
6. Rodar inferências
7. Avaliar e refinar

Definição do objetivo

- É necessário definir o que se deseja prever, para então definir dados a serem coletados e qual arquitetura de modelo utilizar.
- No exemplo da máquina, é um problema de classificação. Normal ou anormal
- No projeto do ar condicionado, saída é um valor de temperatura do ar condicionado? Confortável ou não confortável?

Coleção de Dados

- Este processo pode ser dividido em três etapas:
  - Seleção dos dados
  - Coleta de dados
  - Rotulação dos dados

Coleção de Dados - Seleção dos dados

- Grandes volume e variedade de dados
  - Máquina - temperatura da máquina e tipo de comida servida no almoço.
  - Treinamento para ignorar dados inúteis, porém análise de domínio é importante.
  - Experimentação e combinar diferentes dados.
  - Máquina - taxa de produção, temperatura e vibração.
  - Ar condicionado - temperatura ambiente?

Coleção de Dados - Coleta dos dados

- Quantidade de dados: complexidade das relações entre variáveis, ruído e distinção de classes.
- Quanto mais dados melhor!
- Dados precisam ser representativos das condições: diversidade de condições é importante
- Série temporal com diferentes taxas de aquisição.

Coleção de Dados - Coleta dos dados

- Exemplo da máquina:
  - Temperatura a cada minuto
  - Taxa de produção a cada hora
  - Nível de vibração a cada segundo
- Ar condicionado?
  - Quais variáveis e que taxa?

Coleção de Dados - Rotulando os dados

- Além da coleta é preciso indicar ao modelo o estado da saída daqueles dados.
- No caso da máquina determinamos se cada conjunto de dados representa um estado normal ou anormal.

Data source	Interval	Sample reading
Rate of production	Once every 2 minutes	100 units
Temperature	Once every minute	30°C
Vibration (% of typical)	Once every 10 seconds	23%
Label ("normal" or "abnormal")	Once every 10 seconds	normal

Projetando Modelo de Arquitetura

- Diversidade enorme de arquiteturas.
  - Literatura de arquiteturas para problemas específicos
- Restrições do dispositivo alvo!
  - Número de neurônios, ligações.
  - Dispositivos com aceleradores
- Treina-se um modelo com poucos níveis de neurônios, refinando a arquitetura em um processo iterativo.

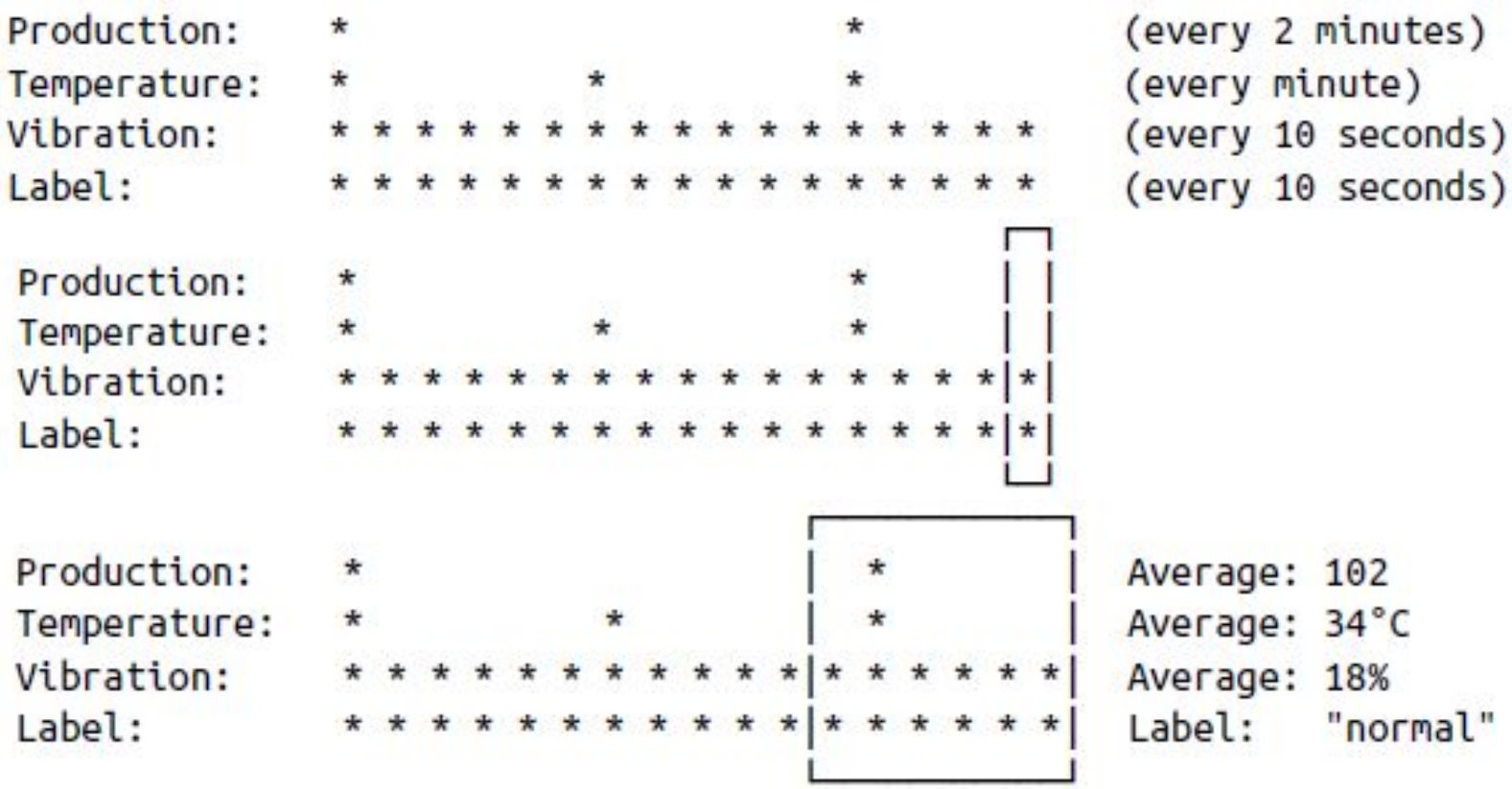
Projetando Modelo de Arquitetura

- Modelos de aprendizado de máquina recebem entradas e geram saídas na forma de tensores. De forma simples, um tensor é essencialmente uma lista que pode conter números ou outros tensores, algo similiar a um *array*.
- Estrutura do tensor tem forma e dimensão.
- Vetor, matriz, dimensões maiores e escalar.

Projetando Modelo de Arquitetura: features

- O termo feature refere-se a um tipo particular de informações no qual o modelo é treinado.
- Diferentes tipos de modelo são treinados com diferentes tipos de feature.
- Modelo pode receber um escalar ou então uma matriz para processar uma imagem e mais outros dados para formar um conjunto

Projetando Modelo de Arquitetura: Janelamento



Projetando Modelo de Arquitetura: normalização

- Os três dados sem o rótulo são as features
  - [102 34 .18]

```
Temperature series:
[108 104 102 103 102]

Mean:
103.8

Normalized values, calculated by subtracting 103.8 from each temperature:
[ 4.2 0.2 -1.8 -0.8 -1.8 ]
```

```
Original 8-bit values:
[[255 175 30]
 [0   45  24]
 [130 192 87]]
```

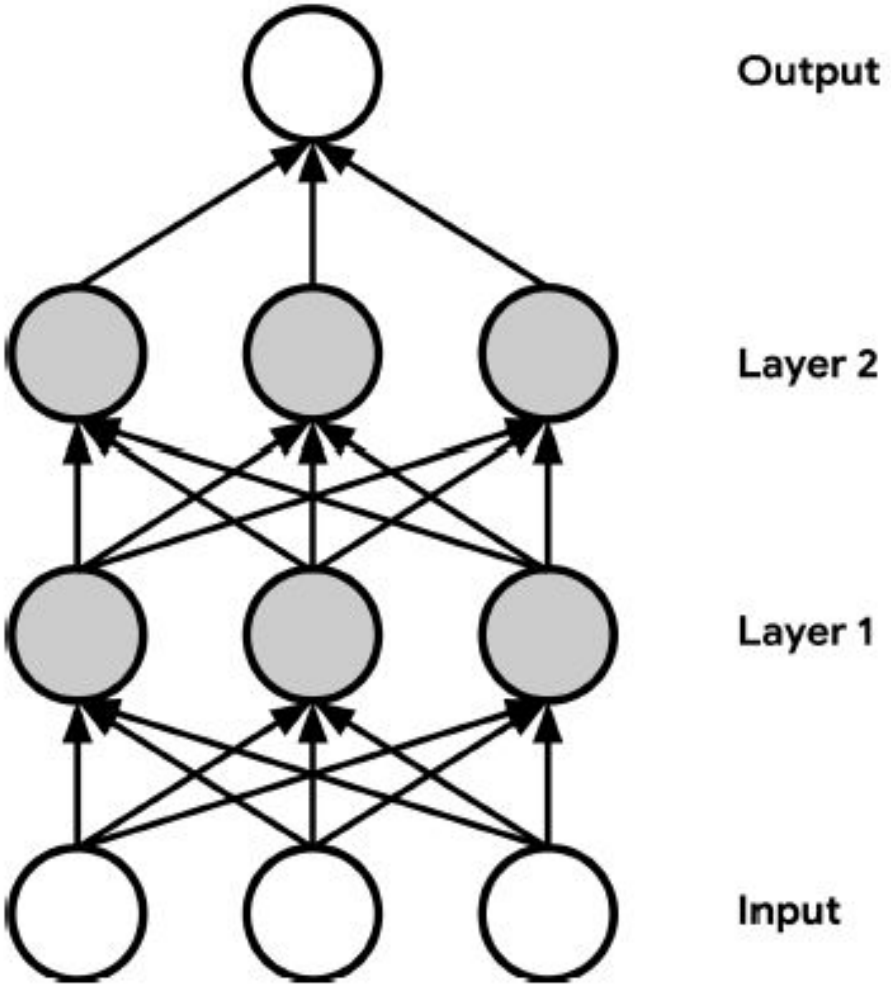
```
Normalized values:
[[1.         0.68627451 0.11764706]
 [0.         0.17647059 0.09411765]
 [0.50980392 0.75294118 0.34117647]]
```

Treinando o Modelo

- Treinamento é o processo pelo qual o modelo aprende a produzir a saída correta para um determinado conjunto de dados de entrada.
- Esse processo envolve alimentar o modelo com dados e fazer pequenos ajustes até que consiga-se as melhores previsões possíveis.
- Um modelo é uma rede de neurônios simulados representados por arrays de números arranjados em camadas.

Treinando o Modelo

- Esses números são conhecidos como pesos and biases, ou coletivamente como parâmetros de rede.
- Dados de entrada são transformados por sucessivas operações matemáticas



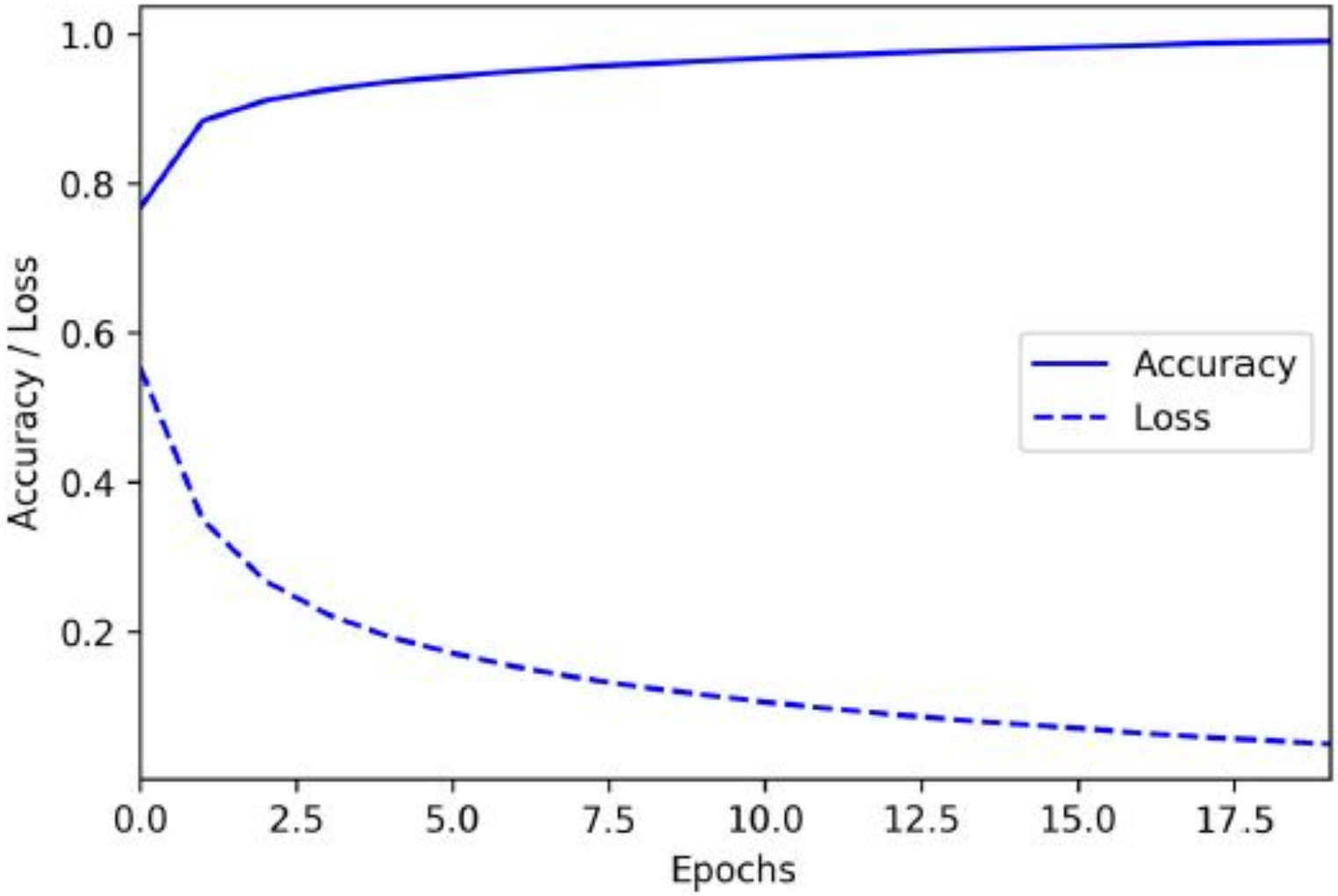
Treinando o Modelo

- Pesos iniciam aleatoriamente e biases com 0.
- O algoritmo backpropagation ajusta os pesos e bias incrementalmente de forma a saída do modelo se ajustando até se aproximar do valor desejado.
- O treinamento, que é medido em epochs (iterações) continua até
- Training, which is measured in epochs (meaning iterations) até que não haja incremento mais na precisão do modelo.

Treinando o Modelo

- Após a convergência do modelo, avaliado através de métricas de desempenho como perda e acurácia.
- Métrica de perda gera uma estimativa numérica de quão longe o modelo está de gerar a saída esperada.
- A métrica acurácia indica o percentual de vezes que o modelo acerta a previsão
- Modelo perfeito: perda de 0.0 e acurácia de 100%

Treinando o Modelo



Treinando o Modelo: Underfitting e overfitting

- Quando um modelo está underfit, ele ainda não é capaz de aprender uma representação significativa dos padrões para uma boa previsão.
- Causado por uma variedade de motivos, mais comum é a arquitetura é pequena demais para capturar a complexidade do sistema ou não há dados suficientes para o treinamento.

Treinando o Modelo: Underfitting e overfitting

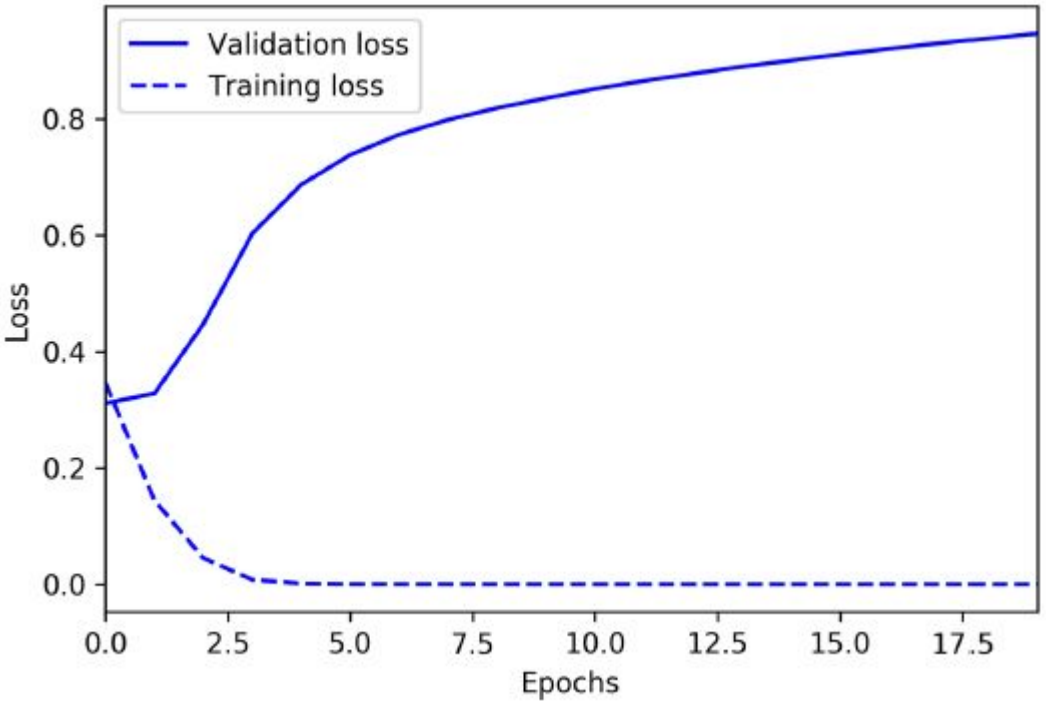
- Quando um modelo está overfit, ele decodificou os dados muito bem, a partir dos dados de entrada ele irá predizer a saída, porém não é capaz de generalizar o seu aprendizado.
- Normalmente acontece porque o modelo conseguiu memorizar todos os conjuntos de entrada ou encontrou um atalho nos dados de treinamento, mas não no mundo real.

Treinando o Modelo: Underfitting e overfitting

- Exemplo de conjunto de dados problemático:
  - Id de cães e gatos, fotos externas e internas.
- Resolver o overfitting através da diminuição do tamanho do modelo
- Métodos como regularization e o data augmentation.

Treinamento, validação e testes

- Recorte de 60% dos dados para treinamento, 20% para validação e 20% para testes.
- Validação reduz overfitting



Convertendo o Modelo

- TensorFlow cria e testa os modelos.
- TensorFlow é essencialmente um conjunto de instruções que dizem ao interpretador como transformar os dados de forma a produzir a saída.
- Quando usarmos o modelo feito, carrega-se ele na memória e executa-se usando o interpretador de TensorFlow.

Convertendo o Modelo

- Interpretador de TensorFlow é feito para hardware robustos e potentes!
- TensorFlow Lite é um interpretador e ferramentas necessárias para rodar os modelos em dispositivos menores e de baixo consumo.
- TensorFlow Lite Converter e quantize

Rodando a Inferência

- Tensor-Flow Lite for Microcontrollers C++ library e código!
- Código para capturar os dados brutos do sensor normalizar e enviar para o modelo.
- Necessidade de médias do valor de saída

Normal score	Abnormal score	Explanation
0.1	0.9	High confidence in an abnormal state
0.9	0.1	High confidence in a normal state
0.7	0.3	Slight confidence in a normal state
0.49	0.51	Inconclusive result, since neither state is significantly ahead

Avaliação e Refinamento

- Resultado pode não ser o esperado!
- Problemas com os dados (overfitting!) ou até mesmo com o hardware.
- Realizar algumas iterações do processo completo.

Machine Learning Toolchain

- Python + Jupyter notebooks
- Google Colaboratory
- TensorFlow + Keras

Exemplos

- https://colab.research.google.com/github/tensorflow/tflite-micro/blob/main/tensorflow/lite/micro/examples/hello_world/train/train_hello_world_model.ipynb
- https://www.survivingwithandroid.com/arduino-machine-learning-tensorflow-lite/
- https://www.tensorflow.org/lite/microcontrollers/get_started_low_level
- https://mirzafahad.github.io/2020-06-26-tflite-stm32-part3/