Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Estatística
Programa de Especialização em *Data Science* e *Big Data*

Joice Pereira Costa

Reconhecimento de doenças em plantações com Machine Learning e sistemas embarcados

Curitiba
2019

Joice Pereira Costa

Reconhecimento de doenças em plantações com Machine Learning e sistemas embarcados

Monografia apresentada ao Programa de Especialização em *Data Science* e *Big Data* da Universidade Federal do Paraná como requisito parcial para a obtenção do grau de especialista.

Orientador: Prof. Marco Alves Zanata Alves

Curitiba
2019

Reconhecimento de doenças em plantações com Machine Learning e sistemas embarcados

Joice Pereira Costa

Universidade Federal do Paraná, Curitiba PR, Brasil

Resumo

Algodão é a fibra natural mais popular no mundo, tendo uma produção mundial em torno de 25 milhões de toneladas todos os anos. O rápido e correto diagnóstico de doenças em plantações é essencial para evitar a propagação de muitas mazelas, o que pode causar perdas na produção e consequentemente perdas financeiras. Para minimizar esses problemas e se beneficiando das novas tecnologias, a agricultura de precisão surgiu por volta dos anos 2000 para otimizar o trabalho em fazendas através da análise de dados de sensores. Hoje em dia, com os avanços tecnológicos de sensores, recursos computacionais e o desenvolvimento de uma base de doenças em plantas confiável, é possível usar algoritmos de *Machine Learning (ML)* para avaliar as condições das plantações, tornando possível, por exemplo, a captura de imagens e o uso de algoritmos de reconhecimento de padrões para identificar múltiplas anomalias, bem como doenças. Neste contexto, este trabalho apresenta o desenvolvimento de dois modelos de ML capazes de reconhecer doenças em plantações de algodão. Os modelos de classificação desenvolvidos foram migrados nos sistemas embarcados Raspberry Pi modelos 3B+ e 2B, usando as bibliotecas de ML do Python para a implementação dos algoritmos de *Multi layer perceptron (MLP)* e *Random Forest (RF)*. Os modelos foram treinados e ajustados em um servidor de alto desempenho utilizando com entrada a base de imagens Digipathos, desenvolvida pela Empresa Brasileira de Pesquisa Agropecuária (Embrapa), composta por um conjunto de imagens de diversas doenças em plantações. Os resultados obtidos mostram que, mesmo com recursos limitados de memória e processamento, o Raspberry Pi 3B+ foi capaz de classificar as doenças em folhas de algodão catalogadas com uma acurácia aproximada de 93%, processando a uma taxa de 16,85 Frames por segundo (FPS) para o modelo MLP, e de acurácia de 97% com taxa de 48,65 FPS para o modelo RF. Enquanto que o Raspberry Pi 2B foi capaz de classificar as mesmas doenças com a mesma acurácia, porém com uma taxa de 1,51 FPS para o modelo MLP, e de 6,55 FPS para o modelo RF.

Palavras-chave: *Machine Learning*, Sistemas embarcados, Detecção de doenças em plantas, Reconhecimento de padrões

Abstract

Cotton is the world's most popular natural fiber, and every year the world produces around 25 million tonnes of cotton. In this way, the prompt and correct diagnosis of crop diseases in plantations is essential to avoid its propagation across the whole plantation, causing financial and production losses. In order to overcome those issues and taking advantage of new technologies, precision agriculture has emerged around the year 2000 and was employed to optimize the work in farms by analyzing data from multiple sensors. Nowadays, with technology advances in sensors, and computing resources, and with the development of reliable plant diseases databases it is possible to use ML applications to evaluate crop conditions, making it possible to capture images in real time and use pattern recognition algorithms to identify anomalies in it, also identifying multiple diseases. In this way, we developed two ML models which is able to recognize diseases in cotton planting. Our classification models were developed to be embedded on the Raspberry Pi boards model 3B+ and 2B, using Python ML libraries to implement MLP and RF algorithms. The models were trained and tested on a high-end server using as input set the Digipathos database, which was built by Brazilian Agricultural Research Corporation (Embrapa) and contains a set of images of different crops diseases. Our results show that, even with the limited memory and processing resources, the Raspberry Pi 3B+ was able to identify the Cotton plant diseases with an accuracy of approximately 93% performing 16.85 FPS when using MLP, and 97% with 48.65 FPS using RF. Meanwhile, the Raspberry Pi 2B presented the same accuracy, but performing 1.51 FPS when using MLP, and 6.55 FPS using RF.

Keywords: *Machine Learning, embedded systems, detection of plant diseases, pattern recognition*

1. Introdução

O algodão é a fibra natural mais popular do mundo. A produção mundial de algodão gira em torno de 25 milhões de toneladas todos os anos. A produção agrícola do algodão depende de correto cultivo, boas condições climáticas e constante inspeção de pragas para atingir bons resultados. Em torno de 80 a 90% das doenças em algodoeiros são em suas folhas, sendo causadas principalmente por fungos [**?**].

A agricultura de precisão está se tornando realidade com o uso de elementos tecnológicos como Veículos aéreo não tripulado (VANTs), dispositivos móveis e sensores sem fio. Entretanto, os agricultores ainda têm muitas perdas financeiras causadas por doenças e pragas nas plantações. A variedade de doenças que podem afetar as plantações é grande e seus sintomas podem variar de acordo com a severidade, condições climáticas e o estágio de desenvolvimento [**?**]. O correto diagnóstico geralmente não é uma tarefa trivial, sendo comumente realizado visualmente por especialistas, podendo ter influência de fatores subjetivos. Outra forma de diagnóstico pode envolver análises laboratoriais, as quais podem consumir um tempo considerável, dificultando uma ação rápida e alto custo [**?**]. Nesse sentido, é essencial o rápido e correto disgnóstico para evitar grandes perdas na produção e consequentemente financeiras.

Na área de ciência da computação, as técnicas de ML originalmente surgiram como uma sub área da Inteligência Artifical (IA), a qual foi desenvolvida por volta dos anos 40. Graças aos avanços tecnológicos e ao desenvolvimento de *frameworks* de linguagens alto nível, as técnicas de ML se tornaram populares podendo ser utilizadas em computadores convencionais[**?**]. Hoje em dia existem muitos artigos utilizando algoritmos de ML para solucionar problemas nas mais diversas áreas, e acreditamos que seu uso no agronegócio pode ser altamente benéfico para o reconhecimento de doenças em plantas. Apesar do reconhecimento de doenças em plantas ser uma atividade comum na agricultura, existem muitos desafios associados, como por exemplo a subjetividade envolvida na avaliação visual [**?**]. Existem, também, muitos desafios no uso de processos automáticos de reconhecimento de imagens, como a presença de fundos de imagem complexos, fronteiras dos sintomas de difícil delimitação, condições ambientais que podem impactar a captura da imagem, entre outros [**?**].

A base de dados Digipathos foi criada pela Embrapa para facilitar o desenvolvimento de processos automáticos de detecção de doenças em plantas, contendo um conjunto de imagens de diferentes culturas afetadas por diferentes doenças. A Digipathos é composta pela base original, chamada Plant Disease Database (PDDB), e pela Base expandida (XDB), sendo resultado de anos de esforço dos pesquisadores, estagiários da Embrapa e estudantes. A primeira (PDDB) possui 2.326 imagens de 171 doenças que afetam 21 espécies de plantas. A segunda (XDB) possui 46.513 imagens e foi criada a partir da base original através do recorte das imagens, contendo imagens isoladas das áreas danificadas das folhas. Esta segunda abordagem viabiliza o uso de técnicas poderosas como *deep learning*. Para garantir contraste com a área danificada, para todas as imagens subdivididas, o material saudável ocupa pelo menos 20% da imagem. Fungos são a causa raiz de muitas doenças, representando 77% da base de dados. Em torno de 30% das imagens da PDDB foram obtidas nas plantações, e o restante foi obtido em ambiente controlado [**?**].

Neste trabalho foi avaliado o uso de técnicas de ML para a classificação de doenças em plantas, baseado nos sintomas, executando em um hardware embarcado. Foram desenvolvidos dois modelos de ML que são capazes de reconhecer doenças em plantações de algodão. Os modelos desenvolvidos foram então executados nos dispositivos Raspberry Pi 3B+ e Raspberry Pi 2B, utilizando as bibliotecas de ML do Python para a implementação de algoritmos de MLP e RF. Os modelos foram treinados e testados em um servidor de alto desempenho, tendo como entrada as imagens de sintomas de doenças em folhas de algodão provenientes da base de dados Digipathos.

Boa parte dos trabalhos anteriores [**?**] [**?**] focavam-se principalmente no diagnóstico de folhas de algodão usando hardware tradicional. Entretanto, neste trabalho o foco está na avaliação do uso de tais técnicas em sistemas embarcados, onde a energia e capacidade de processamento são limitados. Alguns sistemas embarcados permitem o acoplamento de uma câmera, que pode ser utilizada para a captura de imagens em plantações. Sendo dispensável o envio de uma amostra a um laboratório para o diagnóstico. Neste sentido, está sendo proposto o uso de sistemas embarcados para a reconhecimento de doenças em plantas *in loco*. Onde, para a classificação da doença, seriam utilizados algoritmos de ML. Levando em consideração que sistemas embarcados têm limitações em recursos de energia e processamento, os modelos foram treinados em um servidor de alto desempenho e depois migrados para

o sistema embarcado. Essa abordagem pode facilitar o uso de sistemas embarcados no campo, sem o uso de internet, para obter uma rápida resposta para o problema de identificação de doenças em plantações.

Este trabalho é estruturado como segue. A seção **??** aborda alguns trabalhos relacionados a sistemas embarcados e reconhecimento de padrões. A seção **??** aborda os passos realizados para o desenvolvimento do trabalho, desde a manipulação da base de dados ao desenvolvimento dos algoritmos e configuração do sistema embarcado. Na seção **??** são apresentados os resultados obtidos e algumas reflexões a respeito dos valores observados são feitas. Finalmente, na seção **??** são apresentadas as conclusões e sugestões a trabalhos futuros.

2. Trabalhos relacionados

Hoje em dia, com os recursos computacionais disponíveis, diversas áreas estão aplicando soluções de ML, como no caso dos carros autônomos [**?**] [**?**] e na classificação de doenças [**?**] [**?**] [**?**] [**?**]. Entretanto, esses trabalhos são todos baseados em servidores sofisticados com alto poder de processamento ou sistemas Circuitos integrados de aplicação específica (CIAEs).

No estado-da-arte, poucos trabalhos usam sistemas embarcados para execução de algoritmos de ML ou analisam o desempenho de tais sistemas durante a execução dos modelos. Por exemplo, [**?**] também utiliza ML para a detecção de doenças em folhas de Citrus. Para tal, eles utilizaram uma câmera de alta qualidade para capturar as imagens das folhas e segmenta-las com o uso do algoritmo K-Means, e para a identificação e classificação da doença foi utilizado um algoritmo de SVM. Entretanto, este trabalho foi implementado em um servidor de alto desempenho, não permitindo a classificação das imagens *in loco* e sem internet. Metodologia similar foi usada nos trabalhos relacionados [**?**] [**?**], os quais detectam doenças no algodão, utilizando *desktops* ou servidores de *rack*.

Considerando o uso de hardware embarcado, [**?**] propõe um algoritmo para reconhecer gestos das mãos em tempo-real com um sistema embarcado ARM Cortex-A9 com CPU quad-core CPU. Para tal, eles capturaram 1.350 imagens de gestos com as mãos de 15 pessoas e usaram um Modelo de Misturas de Gaussianas (GMM) para classificar os gestos. Com o mesmo conceito, [**?**] desenvolveu um trabalho com classificação de planta do pé. Eles utilizaram Raspberry Pi em uma estrutura desenvolvida para capturar imagens da planta do pé para classifica-lo em tempo-real como normal, chato ou arcado. Ambos trabalhos foram desenvolvidas usando sistemas embarcados, entretanto, eles não utilizaram modelos de ML para a classificação.

Finalmente, [**?**] utiliza novas técnicas baseadas em anomalia para classificação de pacotes utilizando algoritmos de ML. Eles analisam tráfego de dados, neste caso, pacotes de rede com RF, Naive Bayes e kNN, classificando os pacotes em normal ou ataque, onde o objetivo principal foi comparar a acurácia, eficiência energética e tempo de execução da extração de características e classificação com ML. Nosso método difere por fazer classificação de imagens usando hardware de propósito geral.

3. Fundamentos e metodologia

Nessa seção iremos apresentar os fundamentos sobre as imagens utilizadas, os detalhes da modelagem dos classificadores e também a configuração dos sistemas embarcados utilizados.

3.1. Seleção da cultura

Como o foco deste trabalho consiste na avaliação da possibilidade de usar hardware embarcado para identificar doenças em plantas, foi escolhida apenas uma cultura para a análise. Desta forma, após análise da base de imagens, a cultura de algodão foi escolhida levando em consideração os seguintes critérios: Maior número de fotos disponíveis; Maior abrangência das doenças catalogadas para a cultura.

A base PDDB não foi utilizada pois necessitaria de processamento adicional, visto que as fotos mostram a folha inteira e em muitos casos a área afetada representa uma pequena proporção da imagem. Já a base XDB tem foco na área danificada, tornando-se mais adequadas ao uso em algoritmos de ML. Além disso, a base XDB é pelo menos quatro vezes maior que a PDDB. A Tabela **??** mostra a quantidade de elementos da base de algodão para cada uma das doenças catalogadas: Mela (*Rhizoctonia solani*), Mancha-de-Mirotécio (*Myrothecium roridum*), Fusariose (*Fusarium wilt*) e Ramulária (*Ramularia areola*).

3.2. Ajustes na base de fotos

O repositório Digipathos é composto de arquivos compactados, onde cada arquivo compactado representa uma doença. Para o desenvolvimento de um modelo supervisionado, foi necessário adicionar rótulos para

Tabela 1: Imagens de algodão na base Digipathos.

Doenças do Algodão	Número de imagens PDDB	Número de imagens XDB
Rhizoctonia solani	32	166
Myrothecium roridum	27	146
Fusarium wilt	29	0
Ramularia areola	36	1712

cada foto, baseado no agrupamento disponibilizado na base Digipathos.

Além de rótulos, também foram necessários alguns ajustes nas imagens para atingir bons resultados. Primeiramente, para evitar desbalanceamento da base, algumas fotos foram aleatoriamente removidas da base de algodão e, depois do balanceamento de classes da XDB, a base ficou com 452 fotos. A motivação do balanceamento foi para minimizar a chance de alguma classe acabar sendo ignorada pelo modelo [**?**], uma vez que a classe Ramulária possui mais de 10 vezes o número de elementos em comparação às outras 2 classes. A partir da base de algodão balanceada, 70% foi utilizada para treinar o modelo, 20% para validação e 10% foi separado para testes. Todas a fotos foram redimensionadas para 28x28 e normalizadas como parte do fluxo de processamento.

3.3. Desenvolvimento dos modelos de ML

Como a linguagem Python possui vários pacotes que atualmente são comumente utilizados em aplicações de ML, esta foi a linguagem escolhida para o desenvolvimento dos modelos MLP e RF. Os seguintes pacotes foram utilizados para o desenvolvimento:

- **Keras 2.1.3**: Biblioteca de alto nível para redes neurais.
- **Tensorflow 1.1.0**: Plataforma open source para ML, fornecendo ferramentas flexíveis, bibliotecas e uma comunidade de pesquisadores e desenvolvedores.
- **opencv-python-headless**: Interface de Programação de Aplicativos (API) Python do OpenCV.
- **Pandas**: Biblioteca que proporciona estruturas de dados voltadas para a análise e manipulação de dados.
- **Scikit-Learn**: Biblioteca que disponibiliza ferramentas eficientes para mineração e análise de dados.

O modelo MLP foi desenvolvido usando o pacote Keras e para seu ajuste e treino os seguintes parâmetros foram variados até obter uma acurácia satisfatória, onde os melhores parâmetros obtidos foram:

- **Número de camadas** = 3: Quantidade de camadas de neurônios totalmente conectadas.
- **Número de neurônios em cada camada** = (64,32,3): Quantidade de perceptrons usados em cada um das camadas. Cada perceptron fornece uma saída baseada na entrada, pesos e bias.
- **Função de perda** = *categorical cross-entropy*: Função objetivo, que busca minimizar a função de custo. Representa a diferença entre a saída do modelo e a saída real.
- ***Optimizer*** = *adam*: Algoritmo de otimização usado para atualizar os pesos.
- **Função de ativação** = *softmax*: Função utilizada para definir a saída de acordo com a entrada.
- ***batch_size*** = 40: Número de exemplares que serão utilizados antes de atualizar os parâmetros internos.
- **Épocas** = 30: Número de iterações sobre todo o *data-set* fornecido.

O modelo MLP foi desenvolvido considerando uma camada de entrada com 64 neurônios, uma camada escondida de 32 neurônios onde *Rectified Linear Unit (ReLU)* é a função de ativação (a qual normaliza a entrada para valores positivos), e uma camada de saída com 3 neurônios, os quais representam as 3 doenças catalogadas na base algodão XDB. Para a saída foi utilizada a função de ativação *Softmax*, a qual normaliza a entrada da camada para uma distribuição de probabilidade. Para treinar a rede neural foram necessárias 30 épocas, com um *batch size* de 40 exemplares. A representação gráfica da rede neural desenvolvida é ilustrada na Figura **??**.

Levando em consideração que redes neurais são algoritmos complexos para executar em um sistema embarcado, um modelo de RF também foi desenvolvido pois geralmente requer menor poder computacional e consequentemente menos energia. Considerando que energia é um dos fatores cruciais em sistemas embarcados, o modelo RF pode ser uma excelente solução [**?**]. Para ajustar o modelo RF, o módulo *GridSearchCV* do pacote *Scikit-Learn* foi usado para identificar os melhores parâmetros. Os parâmetros ajustados para o RF são descritos abaixo:

- ***random_state*** = 201: Semente usada para a geração de números aleatórios.

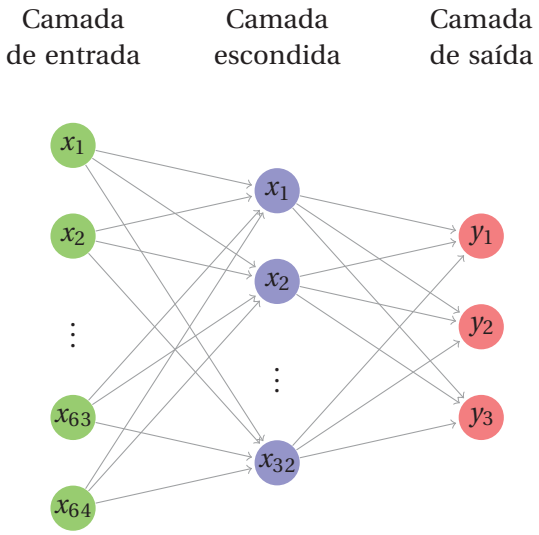

Figura 1: Modelo MLP criado para a base algodão XDB

- ***criterion*** = entropy: Função para medir a qualidade de um split. Valores aceitos são *"gini"* e *"entropy"*.
- ***max_depth*** = 10: Profundidade máxima da árvore, limitando o crescimento da árvore.
- ***n_estimators*** = 50: Número de árvores na floresta.
- ***max_features*** = auto: Número de características a serem consideradas para a busca do melhor *split*.
- ***min_samples_leaf*** = 2: Número mínimo de exemplares que são necessários em um nodo folha.

3.4. Configuração do sistema embarcado

Para a avaliação foco deste trabalho, foi escolhido o Raspberry nas versões Pi 3B+ e 2B, uma vez que apresentam muitas facilidades, como a possibilidade de uso da linguagem Python, conexão com internet, usada para instalar os pacotes necessários, e finalmente ambas plataformas possuem 1 GB de memória RAM.

Para poder utilizar os dois modelos de Raspberry, o sistema operacional Raspbian NOOBS v.3.0.1 foi instalado, com kernel v.4.14 e Python v.3.5.2. Além disso, os seguintes pacotes e bibliotecas foram instalados: Tensorflow 1.8.0, Keras 2.2.4, Scikit-Learn, Pandas, opencv-python-headless e, devido à dependências de pacotes, se fez necessária a instalação de: libatlas-base-dev, h5py, libhdf5-dev, libjasper1.

3.5. Execução de algoritmos de ML

Os algoritmos de ML, MLP e RF, foram treinados e/ou ajustados em um servidor de alto desempenho dotado de dois processadores Intel Xeon 4114 Silver. O modelo MLP foi treinado com um grupo de 305 imagens e validado com outro grupo de 102 imagens, ambas da base XDB. Os parâmetros do modelo foram ajustados através da variação do número de camadas escondidas, número de neurônios por camada, função de ativação, otimizador e função de perda. O modelo final foi construído com os parâmetros que apresentaram a melhor acurácia observada, e posteriormente exportado no formato Notação de Objetos JavaScript (JSON) e salvo em um arquivo. Os pesos do melhor modelo também foram exportados e salvos em um arquivo separado.

Após a fase de treinamento e ajustes, o modelo treinado foi migrado e executado no Raspberry Pi 3B+ e no Raspberry Pi 2B. Um outro programa Python foi desenvolvido para carregar o modelo e seus pesos nos dispositivos Raspberry para testá-lo com a base de teste com um total de 45 imagens XDB, as quais não foram utilizadas nem para o treino e nem para a validação. Para fins de comparação, este mesmo programa foi executado, utilizando os mesmos dados, modelo e pesos, no servidor de alta performance.

Para o RF não foi possível utilizar a mesma abordagem do MLP, no caso, carregar no Raspberry o modelo já treinado. Isso deve-se ao fato de o servidor ter um sistema operacional de 64 bits e o Raspbian ser de 32 bits. Para modelos do tipo árvore não existe um pacote para exportar o modelo de uma forma que seja compatível entre 32 e 64 bits, por exemplo JSON. Para que fosse possível fazer a comparação de desempenho entre as plataformas, foi necessário ajustar o RF no servidor para encontrar os melhores parâmetros. Após o ajuste, os mesmos valores de parâmetros foram utilizados no Raspberry para construir a árvore a partir da base de treino, onde as 305 fotos de treinamento foram armazenadas em um dispositivo de memória adicional (cartão SD) e conectado ao Raspberry Pi 3B+ e também ao Raspberry Pi 2B. Então o mesmo programa de construção do modelo RF foi executado em cada Raspberry usando as fotos de treinamento para a construção do classificador RF. O modelo RF resultante foi exportado através do pacote *pickle* e utilizado para a classificação da base de testes.

4. Resultados e discussões

Nesta seção são apresentados os resultados obtidos na execução dos modelos de ML no servidor de alta performance, bem como nos sistemas Raspberry Pi. As métricas utilizadas para as comparações foram:

- **Acurácia do modelo**: Razão entre as predições corretas e o total de elementos na base de teste.
- **Desempenho em (FPS)**: Taxa de FPS, que representa a quantidade de fotos que o dispositivo foi capaz de classificar por segundo.
- **Energia (Joules)**: Energia necessária para o processamento das imagens, no caso, carregá-las para classifica-las em classes com base nas características.

A Tabela **??** apresenta as principais características computacionais das plataformas avaliadas neste trabalho, com a energia obtida de medições usando o contador interno de hardware para o servidor Intel ou as medições reportadas para o Raspeberry Pi 2B e Pi 3B+ [**?**] [**?**]. O consumo de energia apresentado será usado para inferir a energia usada para a execução de cada um dos modelos de ML.

Tabela 2: Características computacionais.

	Servidor	**Rasp. Pi 3B+**	**Rasp. Pi 2B**
Total de núcleos	10 (SMT)	4	4
Microarquitetura	Intel Skylake	ARM Cortex A53	ARM Cortex A53
Maior Cache	13.75 MB	256 KB	512 KB
Memória SDRAM	132GB	1GB	1GB
TDP todos núcleos inativos	20,2W	1,9W	1,1W
TDP único núcleo ocupado	26,9W	2,6W	1,4W
TDP todos núcleos ocupados	85,0W	5,1W	2,1W

Considerando a média do consumo de energia e a taxa de FPS, pode-se facilmente estimar o total de energia consumida. Para fazer tal estimativa, consideraremos que o número total de amostras a serem analisadas corresponderá a 300 fotos. Durante os experimentos, percebe-se que executando os modelos de MLP e RF no servidor, uma única *thread* foi utilizada, enquanto que no Raspberry Pi os 4 cores foram utilizados para ambos os modelos. Levando em consideração estes detalhes, segue a fórmula geral para o cálculo de energia:

$$TDP\ Servidor_{CoreÚnico} * QtdeFotos/FPS\ Servidor$$

e para ambos modelos de Raspberry Pi é:

$$TDP\ Raspberry_{MultiCore} * QtdeFotos/FPS\ Raspberry$$

4.1. Multilayer perceptron (MLP)

Nesta seção são apresentados os resultados obtidos na execução do MLP no servidor e nos sistemas embarcados.

Como é possível ver na Tabela **??**, a acurácia no servidor e nos Raspberry Pi é a mesma, conforme esperado uma vez que foi utilizado o mesmo modelo e a mesma base de dados. Com relação ao FPS, podemos ver que o Raspberry Pi 3B+ conseguiu processar quase 33.2% da quantia processada em 1 segundo pelo servidor. Enquanto que o Raspberry Pi 2B demonstrou um desempenho bem inferior, processando apenas 2,9% da quantia processada em 1 segundo pelo servidor. Com relação a energia, o modelo 3B+ para processar o mesmo conjunto de imagens consumiu apenas 57% da energia utilizada pelo servidor de alto desempenho, o que demonstra uma alta eficiência deste sistema, já no modelo 2B, o consumo de energia total foi de aproximadamente 263% da energia utilizada pelo servidor. Com isso podemos ver que o Raspberry Pi 3B+ é muito mais eficiente, tanto em taxa de processamento quanto em energia, que a versão Raspberry Pi 2B.

Tabela 3: Comparação do modelo MLP, usando XDB.

Dispositivo	**Métrica**	**Valor**
Servidor	Acurácia no teste	93,33%
	FPS no teste	50,768148
	Custo de energia	158,957 J
Raspberry Pi 3B+	Acurácia do teste	93,33%
	FPS no teste	16,854402
	Custo de energia	90,777 J
Raspberry Pi 2B	Acurácia do teste	93.33%
	FPS no teste	1,505349
	Custo de energia	418,507 J

4.2. Random Forest (RF)

Nesta seção são apresentados os resultados obtidos na execução do RF no servidor e nos sistemas embarcados.

Tabela 4: Comparação do modelo RF, usando XDB.

Dispositivo	**Métrica**	**Valor**
Servidor	Acurácia do teste	97,77%
	FPS no teste	79,69356
	Custo de energia	101,262 J
Raspberry Pi 3B+	Acurácia do teste	97,77%
	FPS no teste	48,65485
	Custo de energia	31,445 J
Raspberry Pi 2B	Acurácia do teste	97,77%
	FPS no teste	6,558504
	Custo de energia	96,058 J

Como podemos ver na Tabela **??**, a acurácia no servidor e nos Raspberry é a mesma, conforme esperado. Com relação ao FPS, é possível notar que o Raspberry Pi 3B+ foi capaz de processar em torno de 61,1% da quantidade de fotos que é processada em 1 segundo pelo servidor. Enquanto que o Raspberry Pi 2B foi capaz de processar apenas 8,2% da quantia de fotos que é processada em 1 segundo pelo servidor. Já na comparação do consumo de energia para a classificação de 300 fotos, podemos observar que o Raspberry Pi 3B+ requer apenas 31% do consumo do servidor, o que demonstra uma eficiência maior do que a observada no modelo MLP. Porém, o Raspberry Pi 2B, consumiu 94,8% da energia utilizada pelo servidor. Isso demonstra que a eficiência do modelo 3B+ é superior a do modelo 2B, tanto em energia quanto em taxa de processamento. Podemos ver também que o consumo de energia no Rasperry Pi 2B para a execução do modelo MLP é pelo menos 4 vezes maior que o consumo observado para o modelo RF.

4.3. Discussão

Embora os modelos tenham sido treinados apenas para classificação de doenças do algodão, consideramos que tais resultados podem ser expandidos para outros contextos.

Analisando a performance dos modelos consideramos os dois modelos possíveis de serem utilizados em sistema embarcado. Para o MLP executado no Raspberry Pi 3 B+, alcançamos uma taxa de FPS de metade da taxa de captura da câmera do Raspberry, que é de 30 FPS (considerando o sensor Omnivision OV5647 CMOS 5-MP). Para o RF, obtemos uma taxa 60% maior que a maior taxa da câmera. Com relação ao Raspberry Pi 2B o desempenho foi bem inferior, conseguindo processar 5% da taxa de captura da câmera para o modelo MLP e aproximadamente 22% para o modelo RF. No quesito consumo de energia, o uso de algoritmos mais robustos, como o MLP, no Raspberry Pi 2B pode ser tornar inviável devido a sua ineficiência energética. Já para um algoritmo mais simples, o RF, o dispositivo necessita de aproximadamente 23% da energia necessária para a classificação de 300 fotos no modelo MLP, o que demonstra que o modelo RF é mais viável para este dispositivo num cenário onde o recurso de energia é limitante.

5. Conclusões e trabalhos futuros

Considerando que doenças em plantações ainda geram consideráveis prejuízos na agricultura, e que a correta identificação não é trivial, o rápido e correto diagnóstico pode evitar que a doença se espalhe, reduzindo os prejuízos além de aumentar a segurança alimentar. O foco deste trabalho foi avaliar a viabilidade do uso de hardware embarcado, especificamente o Raspberry Pi 3B+ e 2B, para o reconhecimento de doenças em plantas. Para esta avaliação, foi escolhida a cultura de algodão da base de imagens Digipathos, desenvolvida pela Embrapa, devido ao número de exemplares em cada categoria, sendo quatro categorias catalogadas na Digipathos: Mela , Mancha-do-mirotécio, Fusariose e Ramulária.

Os resultados obtidos nesse trabalho mostram que é possível ter um modelo de reconhecimento de doenças em plantas eficiente, como o desenvolvido para a cultura de algodão, sendo executado em um hardware embarcado. Com ressalvas ao uso de algoritmos mais robustos, como o MLP utilizado, no Raspberry Pi 2B. O uso de hardware embarcado viabiliza a mobilidade da solução a um baixo custo.

Acreditamos que a acurácia do modelo desenvolvido é satisfatória, onde para o modelo MLP ficou em torno de 93% e de aproximadamente 97% para o modelo RF, utilizando a base algodão XDB.

Com relação ao desempenho do Raspberry Pi 3B+, o modelo MLP desenvolvido é capaz de classificar em torno de 16 FPS, enquanto que o modelo RF, que é um algoritmo mais simples, é capaz de classificar em torno de 48 FPS. No que diz respeito ao consumo de energia, o modelo MLP consumiu apenas 57% da energia necessária para o mesmo processamento em um processador de alto desempenho, onde o RF consumiu apenas 31% da energia necessária no servidor. Esses resultados mostram que o modelo RF pode processar três vezes o número de fotos processadas pelo modelo MLP em um segundo, tendo também apresentado uma acurácia levemente maior.

O desempenho do Raspberry Pi 2B foi inferior, se comparado à versão 3B+, sendo capaz de classificar apenas 1,5 FPS para o modelo MLP, enquanto que o modelo RF, que é um algoritmo mais simples, é capaz de classificar em torno de 6 FPS. No que diz respeito ao consumo de energia, o modelo MLP consumiu 263% da energia necessária para o mesmo processamento em um processador de alto desempenho, e o RF consumiu 94% da energia necessária no servidor. Esses resultados mostram que o MLP necessitou de pouco

mais de 4 vezes a quantia de energia utilizada pelo modelo RF para classificar o mesmo conjunto de imagens. Demonstrando um consumo de energia significativamente maior para algoritmos de ML mais robustos, como o algoritmo MLP desenvolvido.

Como sugestão para trabalhos futuros, seria interessante a expansão do modelo, incluindo mais culturas ou até mesmo avaliando a possibilidade de um modelo de reconhecimento de doenças em plantas aplicável para todas as culturas. Neste segundo cenário, seria necessário analisar se os sintomas das doenças possuem características semelhantes em diferentes culturas. Com a expansão do modelo, seria possível disponibilizar uma aplicação robusta de reconhecimento de doenças em plantas executando em um Raspberry Pi 3B+, e consequentemente ser utilizado pelos agricultores.