

Detecção de vagas disponíveis e previsão de ocupação em estacionamentos em ambientes abertos

Felipe Jun Ting Lin
Centro de Informática
UFPE
Recife, Brazil
fjtl2@cin.ufpe.br

Gabriel de Melo Evangelista
Centro de Informática
UFPE
Recife, Brazil
gme@cin.ufpe.br

João Gabriel Vasconcelos
Centro de Informática
UFPE
Recife, Brazil
jgav@cin.ufpe.br

Maria Luísa Lima
Centro de Informática
UFPE
Recife, Brazil
mlll@cin.ufpe.br

I. OBJETIVO

Desenvolver um detector que identifique vagas livres em estacionamentos abertos através das redes neurais convolucionais AlexNet e mAlexNet [2], e da família de modelos de detecção Yolov5. Além disso, desenvolver um preditor da ocupação do estacionamento através das redes neurais recorrentes *Long Short-Term Memory (LSTM)* [5] e *Gated Recurrent Unit (GRU)* [6].

II. JUSTIFICATIVA

Encontrar uma vaga livre em estacionamentos de grandes cidades frequentemente se torna exaustivo, consumindo tempo e dinheiro consideráveis. Tecnologias automatizadas capazes de detectar a ocupação de vagas de estacionamento poderiam auxiliar os motoristas que as procuram, informando exatamente onde um local disponível estaria localizado. Além disso, a detecção por imagem permite que câmeras sejam usadas para cobrir grandes áreas, diminuindo custos de, por exemplo, sensores individuais por vaga, que por vezes se encontram danificados, atrapalhando o condutor.

Ademais, a previsão da ocupação do local pode auxiliar o motorista a se programar antes de se deslocar, permitindo que seu dia a dia seja facilitado, ou até permitindo que o administrador do estacionamento tome medidas para mitigar a super lotação do espaço.

III. BASES DE DADOS E ANÁLISE EXPLORATÓRIA

Este estudo utilizará os *datasets* de imagens de estacionamentos PKLot [1], CNRPark, CNRPark-EXT [7] e Utmach. Além disso, para a previsão de ocupação, dois *datasets* serão criados a partir do PKLot e CNRPark-EXT.

1) Detecção de ocupação:

a) *PKLot*: O *dataset* PKLot possui 12417 imagens capturadas em 2 estacionamentos com 3 câmeras (PUC, UFPR04 e UFPR05) e sob condições climáticas variadas. Cada imagem de vaga foi manualmente verificada e classificada de acordo com sua situação e condição climática observada. Na Figura 1 é possível observar um exemplo de estacionamento. Na Tabela II é possível observar a distribuição das vagas ocupadas e livres e na Tabela I é possível observar a distribuição das condições climáticas.



Figura 1: Exemplo de imagem incluída no dataset PKLot

Tabela I: Distribuição de imagens PKLot

Estacionamento	Condição climática	# de imagens
UFPR04 (28 vagas)	Ensolarado	2098
	Nublado	1408
	Chuvoso	285
	Subtotal	3791
UFPR05 (45 vagas)	Ensolarado	2500
	Nublado	1426
	Chuvoso	226
	Subtotal	4152
PUCPR (100 vagas)	Ensolarado	2315
	Nublado	1328
	Chuvoso	831
	Subtotal	4474

b) *CNRPark*: O CNRPark possui 242 imagens capturadas em dias ensolarados por duas câmeras em um estacionamento. Na Figura 2 é possível observar uma imagem de uma das câmeras e na Tabela II é possível observar a distribuição das classes.

c) *CNRPark+EXT*: O CNRPark+EXT possui 4287 imagens capturadas por 9 câmeras em ângulos diferentes de um estacionamento, sob as mesmas condições climáticas do PKLot. Na Figura 3 é possível observar a visão de uma das câmeras e na Tabela II é possível observar a distribuição das classes.

d) *Utmach*: O *dataset* Utmach possui 1000 imagens capturadas em uma variedade de estacionamentos arbitrários que contemplam uma gama diversificada de condições climáticas.



Figura 2: Exemplo de imagem incluída no *dataset* CNRPark

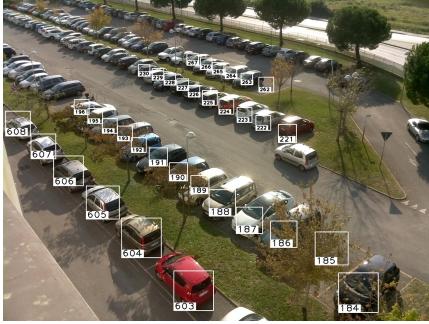


Figura 3: Exemplo de imagem incluída no *dataset* CNRPark+EXT

e posições de câmera. O *dataset*, disponível na plataforma *Roboflow* (framework de desenvolvimento para Visão Computacional que permite um melhor processamento de dados e provê modelagem de técnicas de treinamento variadas) é segmentado por padrão em 800 imagens de treino e 200 imagens de teste, com o diferencial de que a plataforma permite realizar *data augmentation* no banco de dados original a fim de aumentar a generalização do modelo. Em virtude da sua forte integração com o Yolov5, esse *dataset* não foi comparado de forma técnica com os demais bancos comentados nesse documento, apresentando uma análise única. Na Figura 4 é possível observar uma das imagens pertencentes ao dataset Utmach, juntamente com a segmentação de classes através de *bounding-boxes*.

2) *Previsão de Ocupação*: Os *datasets* de imagens não possuem a informação da ocupação total a cada momento, por isso foi necessária a extração manual dessas informações. Através da soma da informação de cada veículo foi possível construir uma tabela com dados relevantes à predição da ocupação, como o ano, mês, dia, hora e minuto, clima, dia da semana e a ocupação. A tabela do PKLot possui dados espaçados de forma não padrão, variando de 1 a 15 minutos, totalizando 12416 linhas. A tabela do CNRPark-EXT possui dados espaçados a cada 5 minutos, totalizando 3416 linhas.

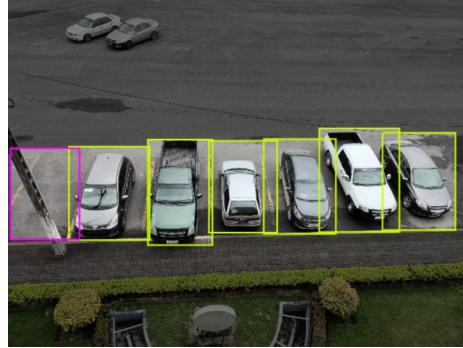


Figura 4: Exemplo de imagem incluída no *dataset* Utmach

Tabela II: Distribuição da Ocupação

Base de Dados	Espaços Livres	Espaços Ocupados
CNRPark	4181	8403
CNRPark-EXT	65658	79307
PKLot	337780	358119

Tabela III: Dados temporais do CNRPark-EXT

Clima	Ano	Mês	Dia	Hora	Minutos	Dia da Semana	Ocupação
2	2015	7	3	8	5	4	6
2	2015	7	3	8	10	4	7
2	2015	7	3	8	15	4	8
2	2015	7	3	8	20	4	8
2	2015	7	3	8	25	4	9

IV. MODELOS DE APRENDIZAGEM PROFUNDA

Para a detecção da ocupação de vagas, redes neurais convolucionais foram utilizadas pois são capazes de extrair características de imagens de vagas livres e ocupadas. Para a previsão da ocupação do estacionamento, redes neurais recorrentes foram utilizadas pois conseguem capturar características dos dados de ocupação ao longo do tempo.

A. Redes Neurais Convolucionais

Este estudo utilizou duas redes para comparação, as redes AlexNet e mAlexNet.

1) *AlexNet*: A Rede AlexNet [8] é um classificador de 1000 classes que em 2012 foi capaz de vencer o *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) por uma margem de mais de 10% relação ao segundo colocado. Ela possui 8 camadas, sendo as 5 primeiras convolucionais, seguidas por camadas de *max-pooling*, e por fim, 3 camadas totalmente conectadas. A entrada possui dimensões 224 x 244 x 3, totalizando mais de 60 milhões de parâmetros na rede. A arquitetura em detalhes pode ser observada na Figura 5.

2) *mAlexNet*: A Rede mAlexNet [2] é um classificador inspirado no AlexNet que possui 3 camadas convolucionais e 2 camadas totalmente conectadas, incluindo a camada de saída. As duas primeiras camadas convolucionais são seguidas por *Max Pooling*, *Local Response Normalization* (LRN) e função de ativação *ReLU*. A terceira camada convolucional não utiliza LRN. O número de filtros nas camadas de convolucionais e o número de neurônios na camada totalmente conectada

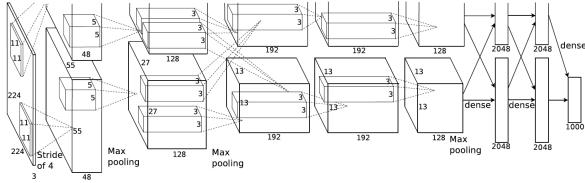


Figura 5: Arquitetura do AlexNet.

são reduzidos para se adequar ao problema, obtendo uma arquitetura com cerca de 1/1340 parâmetros do número de parâmetros do AlexNet. A entrada também possui dimensões 224 x 224 x 3. A arquitetura em detalhes pode ser observada na Figura 6.

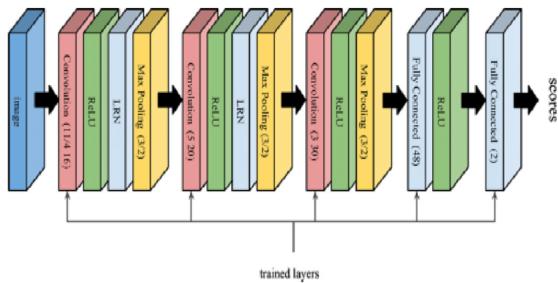


Figura 6: Arquitetura do mAlexNet.

B. Redes Neurais Recorrentes

1) *LSTM: Long Short Term Memory*, ou LSTM, é um tipo de rede neural recorrente (RNN) que é capaz de aprender sequências de longo prazo, conseguindo lembrar de longas sequências por um longo período de tempo. A razão pela sua popularidade se deve ao fato de que onde em uma célula RNN normal, a entrada no *timestamp* e no estado oculto da etapa de tempo anterior seria passada pela camada de ativação para obter um novo estado, no LSTM, a célula recebe a entrada de três estados diferentes: estado da entrada atual, a memória de curto prazo da célula anterior e a memória de longo prazo. Essa arquitetura pode ser visualizada na Figura 7.

Cada célula utiliza os *gates* para regular as informações que serão mantidas ou descartadas na operação de *loop* antes de passar as informações de longo e curto prazo para a próxima célula. Sendo assim, é possível imaginar esses *gates* como filtros que selecionam ou removem informações desnecessárias. Os *gates* utilizados no LSTM são:

- *Input Gate*: Decide quais informações serão armazenadas na memória de longo prazo. Funciona apenas com as informações da entrada atual e da memória de curto prazo da etapa anterior. Logo, filtra as informações de variáveis que não são relevantes.
- *Forget Gate*: Decide quais informações da memória de longo prazo serão mantidas ou descartadas, multiplicando a memória de longo prazo recebida por um vetor de esquecimento gerado pela entrada atual e pela memória curta recebida.

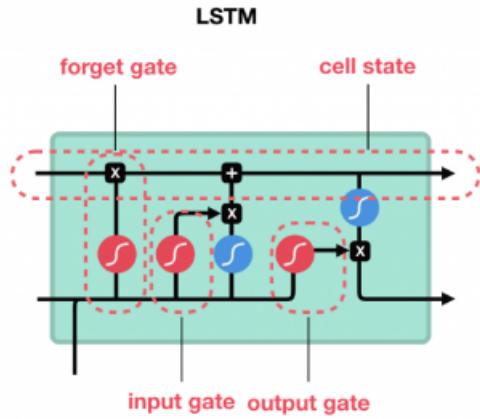


Figura 7: Arquitetura do LSTM

- *Output Gate*: Passa a entrada atual, a memória de curto prazo anterior e a memória de longo prazo para a célula na próxima etapa de tempo, produzindo uma nova memória de curto prazo. A saída do passo de tempo atual também pode ser extraída desse estado oculto.

2) *GRU: Gated Recurrent Unit*, ou GRU, é uma outra variante da arquitetura de rede neural recorrente (RNN) que também usa mecanismos de *gating* para controlar e gerenciar o fluxo de informações entre as células da rede neural. A estrutura do GRU permite capturar de forma adaptativa as dependências de grandes sequências de dados sem descartar as informações de passos anteriores da sequência e também consegue resolver alguns problemas dos tradicionais RNN, como a explosão e a dissipação de gradiente. Essa arquitetura pode ser visualizada na Figura 8.

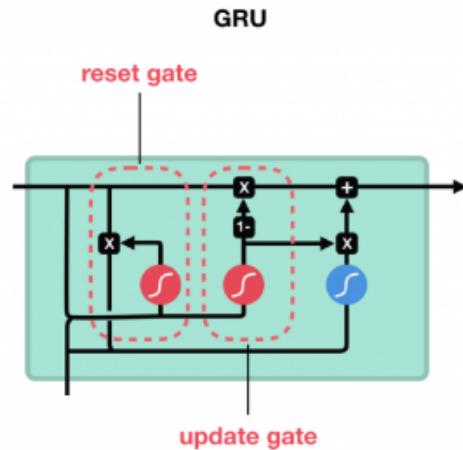


Figura 8: Arquitetura do GRU

Os *gates* utilizados pelo GRU são:

- *Update Gate*: Responsável por determinar a quantidade de informações anteriores que precisam passar para o próximo estado, sendo possível copiar todas as

informações do passado e eliminar o risco da dissipação de gradiente.

- *Reset Gate*: Responsável para decidir quantas informações do passado devem ser esquecidas, podendo determinar se o estado anterior da célula é relevante.

C. Família Yolov5

YOLOv5 é uma família de modelos de detecção escaláveis treinados no dataset COCO, e inclui funcionalidades simples para *Test Time Augmentation* (TTA) (*augmentation* realizada durante a avaliação da performance do modelo e na realização de novas previsões, ensembling de modelo, evolução de hiperparâmetros e exportação para ONNX, CoreML e TFLite - que são *builds* representativas de modelos de aprendizado de máquina).

O COCO, por sua vez, é um dataset em larga escala utilizado para detecção de objetos, segmentação e catalogação, e possui as seguintes features:

- Segmentação de Objetos
- Reconhecimento em contexto
- Segmentação de Superpíxel
- 330K imagens (>200K catalogadas)
- 1.5 milhões de exemplos de objetos
- 80 categorias de objetos
- 91 categorias de coisas
- 5 legendas por imagem
- 250000 pessoas com pontos chave

A esquematização de arquitetura do Yolov5 inclui três partes principais: A espinha dorsal (*Backbone*), o pescoço (*Neck*) e a cabeça (*Head*).

1) *Backbone*: Utiliza a técnica do CSPNet (*Cross Stage Partial Networks* a fim de extrair *features* mais informativas das imagens de entrada. Essa rede respeita a variância dos gradientes através da integração de *feature-maps* do começo ao fim da rede, o que reduz as computações em cerca de 20% relativo à rede equivalente em quesito de acurácia do dataset ImageNet. O CSPNet mostra melhoria significativa no processamento de redes profundas.

2) *Neck*: É principalmente utilizado, com a técnica PANet (que possui a habilidade em preservar informações espaciais de forma acurada, provendo um realce/melhoria no processo de segmentação de instâncias), para gerar pirâmides de features, que ajudam a generalizar bem a escala de objetos, auxiliando na identificação do mesmo objeto em diferentes tamanhos.

3) *Head*: Referente à camada YOLO, é utilizado para a parte da detecção final. Sua principal função é aplicar bounding-boxes de tamanhos variados nas features e gerar um vetor de classes como saídas contendo a probabilidade, *scores* e *bounding-boxes*.

V. EXPERIMENTOS

A. Detecção de Ocupação

1) *Redes Neurais Convolucionais*: Inicialmente haviam sido realizados estudos de treinamento e teste utilizando-se apenas um dos *datasets* por experimento, porém foi observado

que o modelo utilizado não era generalista, apresentando boas métricas apenas para *dataset* de treino. Por exemplo, o modelo mAlexNet treinado no estacionamento UFPR04 apresentou bons resultados apenas nos estacionamentos UFPR04 e PUC, como observado na Tabela IV, onde ambos são da base de dados PKLot.

Tabela IV: Resultados do mAlexNet treinado no UFPR04

Estacionamento	Acurácia	Auc	F1 Score
camera2	0.7066	0.6364	0.7938
camera4	0.7917	0.7744	0.8314
camera6	0.7801	0.7653	0.8207
camera8	0.7636	0.7571	0.8037
PUC	0.9227	0.9260	0.9192
B	0.6973	0.5989	0.7974
UFPR04	0.9713	0.9708	0.9671

Buscando trazer uma maior generalização para o modelo, o *dataset* de treino foi criado buscando-se misturar os dados trazidos na seção III da forma apresentada na Tabela V.

A fim de testar a aplicação em dados da realidade do Centro de Informática (CIn), foi colocado um tripé com uma câmera, como observado na Figura 9, no 4º andar do Bloco E do mesmo, por cerca de 1 hora em modo *timelapse*, resultando em um vídeo de 30 segundos. O vídeo obtido foi utilizado como artefato para inferência da rede treinada.

Tabela V: Distribuição dos dados em treino e teste.

Base de dados		
	CNR-EXT	PKLot
Treino	1, 3, 5, 7, 9	UFPR04, UFPR05
Teste	2, 4, 6, 8	PUC

Este vídeo teve seus frames analisados pela equipe, e o local das vagas foi definido através de uma *bounding-box* para que fossem feitas as inferências dos modelos.

Antes de serem alimentados às redes, os dados foram redimensionados para as dimensões 256 x 256 e então um recorte aleatório de tamanho 224 x 224 foi realizado para adequar o dado à entrada da rede. Além disso, os dados foram normalizados.

Ao treinar as redes nos dados aqui especificados, foi observada a dificuldade de fazê-lo através do *Google Colab*, o que seria mais vantajoso tendo em vista a aceleração de computação provida pela *GPU* do mesmo. Por isso, o treinamento foi realizado em computador local, aumentando o tempo necessário para o mesmo. Ademais, os dados foram amostrados em 10% de cada dataset, para reduzir o custo computacional. Da mesma forma, não foi realizada a validação cruzada de *k-fold* pois isto também levaria a um grande aumento no tempo de treinamento das redes. Em todos os experimentos utilizou-se os seguintes hiperparâmetros para os modelos AlexNet e mAlexNet:

- Número de *Epochs*: 18;
- Taxa de Aprendizagem: 0.01 sendo dividida por 2 a cada 6 *epochs*;



Figura 9: Tripé montado no 4º andar do bloco E do CIn para captura de imagens.

- Tamanho do *Batch*: 64;
- Função de *Loss*: Entropia Cruzada;

Esses hiperparâmetros foram os mesmos utilizados em [2]. As métricas de avaliação utilizadas foram acurácia, área sob a curva (AUC) ROC, que mede quão próximo a curva ROC está da classificação perfeita, e o F1 Score, que mede a média harmônica entre precisão e *recall*.

2) Família Yolov5:

a) *Utmach*: O dataset Utmach foi treinado no Yolov5 com a proporção de 800 imagens de treinamento e 200 imagens de validação, seguindo os seguintes parâmetros:

Tabela VI: Parâmetros do treinamento do Utmach no Yolov5

Epochs	Batch Size	Image Size	Optimizer	Learning Rate
10	16	640x640	SGD	0.01

Os demais parâmetros relevantes foram utilizados segundo a padronização de *pipeline* do Yolov5.

b) *CIn*: O dataset CIn utilizado para treinamento no Yolov5 com o valor de 708 imagens de treinamento e 177 imagens de validação, seguindo os seguintes parâmetros:

Tabela VII: Parâmetros do treinamento do Utmach no Yolov5

Epochs	Batch Size	Image Size	Optimizer	Learning Rate
10	4	1280x1280	SGD	0.01

Os demais parâmetros relevantes foram utilizados segundo a padronização de *pipeline* do Yolov5.

B. Previsão de Ocupação

Nessa seção será demonstrado como foi obtido os resultados da série temporal através das técnicas das redes neurais recorrentes: LSTM (*Long Short Term Memory*) e GRU (*Gated Recurrent Unit*) sobre os *datasets* de CNRPark e PKLot. E devido ao fato que os valores e os parâmetros utilizados em ambos banco de dados foram os mesmos, essa seção mostrará apenas o desenvolvimento do *dataset* de CNRPark e no próximo tópico, os resultados obtidos e as comparações com os dados originais e valores preditos.

Primeiramente, houve a separação das colunas para realizar a predição do modelo. Sendo assim, do dataset original, tabela III, foram usadas como *features* as colunas que indicam o tempo atual *weather*, *year*, *month*, *hour*, *minutes*, *weekday*. Como alvo foi utilizada a coluna *occupancy*, visto que indica a quantidade de vagas disponíveis naquele momento do dia.

Em seguida, foram aplicados 2 métodos de normalização de dados, o *StandardScaler* para as *features* e o *MinMaxScaler* para o *target*. Como o alvo indica um problema de regressão, é importante realizar essa normalização para que não haja um grande crescimento da *loss*. A seguir, foi feita uma divisão do dados de 80/20 entre treino e teste. Por ter sido utilizada a biblioteca *PyTorch*, todos os valores foram convertidos para tensor de forma que o modelo consiga realizar os cálculos do gradiente.

1) *LSTM*: O modelo LSTM foi criado com 3 camadas: uma camada LSTM que recebe os hiperparâmetros número de camadas LSTM, dimensão da entrada, dimensão da camada oculta e a taxa de *dropout*; uma camada totalmente conectada; e por fim, uma camada ReLU. Foi criada também a função de *forward* e a função de inicialização da camada oculta em formato de tupla contendo tanto o estado da célula, quanto o estado oculto.

2) *GRU*: O modelo GRU foi criado de forma similar ao LSTM: uma camada GRU que recebe os hiperparâmetros número de camadas GRU, dimensão da entrada, dimensão da camada oculta e a taxa de *dropout*; uma camada totalmente conectada; e por fim, uma camada ReLU. Foi criada também a função de *forward* e a função de inicialização da camada oculta com apenas o estado oculto.

As variáveis auxiliares, ou hiperparâmetros, definidas acima foram definidas da seguinte forma para o modelo de LSTM e para o modelo de GRU:

Tabela VIII: Parâmetros do treinamento do CNRPark e PUC nos modelos LSTM e GRU

Epochs	Input Dim	Hidden Dim	Output Dim	Learning Rate	Num Layers
1000	6	128	1	0.001	2

A função de *Loss* foi definida, seguida do método de otimização através do *Adam*. Com o objetivo de comparar

o desempenho de ambos os modelos, foi feito um acompanhamento do tempo de treinamento e foi escolhida uma métrica para, eventualmente, equiparar a precisão final de ambos os modelos no conjunto de teste através do sMAPE (*Symmetric Mean Absolute Percentage Error*). Este representa o somatório da diferença absoluta entre o valor predito e o valor original, dividido pela média desses valores, resultando em uma porcentagem que mede a quantidade de erro.

VI. ANÁLISE DOS RESULTADOS

A. Redes Neurais Convolucionais

Os modelos AlexNet e mAlexNet foram treinados e testados com a divisão dos estacionamentos descrita na Tabela V. Os tempos de treinamento e inferência de ambos os modelos divergiram bastante. O modelo AlexNet necessitou de cerca de 6 horas para treinar enquanto que o modelo mAlexNet necessitou de apenas 1 hora. Porém como observa-se nas tabelas IX e X, o AlexNet apresentou performance melhor em todos os estacionamentos testados e em todas as métricas utilizadas.

Tabela IX: Tabela de resultados do AlexNet

AlexNet			
Dataset	Accuracy	Auc	F1 Score
camera2	0.93154	0.94571	0.94756
camera4	0.93844	0.94109	0.94364
camera6	0.94225	0.94255	0.94444
camera8	0.94432	0.94490	0.94707
PUC	0.94931	0.95128	0.94609
B	0.82683	0.84040	0.87842

Tabela X: Tabela de resultados do mAlexNet

mAlexNet			
Dataset	Accuracy	Auc	F1 Score
camera2	0.88264	0.89754	0.90874
camera4	0.89740	0.90315	0.90340
camera6	0.91139	0.91205	0.91400
camera8	0.90818	0.90911	0.91205
PUC	0.93454	0.93746	0.93127
B	0.77535	0.79985	0.83747

Por ter apresentado performance melhor, o vídeo capturado no CIn foi alimentado ao modelo AlexNet e um vídeo foi gerado apresentando a quantidade de vagas livres a cada momento. Na Figura 10 é possível observar um dos momentos.

Como não há rótulos para cada vaga no tempo, não foi possível calcular métricas nesse vídeo, porém é possível observar que em algumas vagas o modelo performa bem enquanto que em outras troca diversas vezes de previsão. Foi criado também um vídeo para o estacionamento da PUC pois este pertence ao conjunto de teste e apresenta os rótulos a cada momento. Durante todo o vídeo, o modelo AlexNet apresentou



Figura 10: Inferência do AlexNet no CIn.

acurácia maior que 80% e em alguns momentos chegou a 100%. Na Figura 11 é possível observar um dos momentos com 100% de acurácia.



Figura 11: Inferência do AlexNet na PUC.

B. Previsão de Ocupação

1) CNRPark-EXT: Foi observado então que os resultados obtidos pelos modelos de LSTM e GRU sobre o dataset de CNRPark-EXT tiveram uma diferença em relação ao tempo gasto para realizar o treino e o desempenho do mesmo, onde o LSTM, devido a sua complexidade, foi mais lento no treinamento (448.35s) mas teve uma acurácia melhor (sMAPE: 3.29%), enquanto que o GRU foi mais rápido (441.74s) mas teve uma acurácia menor (sMAPE: 14.54%). Esses resultados podem ser vistos nas Figuras 12, 13, 14 e 15.

```
Epoch 1000/1000 Done, Total Loss: 0.014302102164776228
Total Time Elapsed: 0.4227840000000356 seconds
Total Training Time: 441.7378369999996 seconds
```

Figura 12: Tempo de execução do treinamento: GRU

```
Epoch 1000/1000 Done, Total Loss: 0.006313035448821371
Total Time Elapsed: 0.46286800000007133 seconds
Total Training Time: 448.3472629999985 seconds
```

Figura 13: Tempo de execução do treinamento: LSTM

```

Evaluation Time: 0.005403000000001157
sMAPE: 14.543408155441284%
/usr/local/lib/python3.7/dist-packages/i
/usr/local/lib/python3.7/dist-packages/i

```

Figura 14: Desempenho (sMAPE): GRU

```

Evaluation Time: 0.00769999999999818
sMAPE: 3.288859501481056%
/usr/local/lib/python3.7/dist-packages/i
/usr/local/lib/python3.7/dist-packages/i

```

Figura 15: Desempenho (sMAPE): LSTM

Por fim, a Figura 16 mostra o gráfico dos dados originais (linha azul) e dos dados preditos (linha verde), no qual é possível observar que os valores obtidos se assemelham com o comportamento do *dataset* real, onde o gráfico da parte superior pertence ao resultado do modelo GRU e o gráfico da parte inferior pertence ao resultado do modelo LSTM:

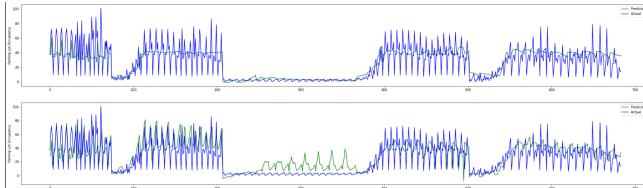


Figura 16: Gráfico GRU e LSTM CNRPark-EXT

2) *PUC*: Como a tabela de informações do PKlot comprehende os 3 estacionamentos: UFPR04, UFPR05 e PUC, utilizou-se apenas o estacionamento PUC, que havia sido utilizado para teste. Foi observado então que os resultados obtidos pelos modelos de LSTM e GRU sobre o *dataset* da PUC teve uma diferença de tempo e desempenho maior nos modelos do que no *dataset* de CNRPark, no qual o LSTM, devido a sua complexidade, continuou sendo mais lento no treinamento (643.61s) mas teve uma acurácia melhor (sMAPE: 24.6%), enquanto que o GRU foi mais rápido no treinamento (612.91s), mas não conseguiu ter um bom desempenho (sMAPE: 91.92%), indicando que o valor do erro foi maior do que o valor real. Esses resultados podem ser vistos nas figuras 17, 18, 19 e 20.

```

Epoch 1000/1000 Done, Total Loss: 0.0017298850713239255
Total Time Elapsed: 0.635508999999564 seconds
Total Training Time: 612.9114239999932 seconds

```

Figura 17: Tempo de execução do treinamento: GRU

```

Epoch 1000/1000 Done, Total Loss: 0.0007085870299710174
Total Time Elapsed: 0.6540140000001884 seconds
Total Training Time: 643.6139950000033 seconds

```

Figura 18: Tempo de execução do treinamento: LSTM

```

Evaluation Time: 0.009298999999828084
sMAPE: 91.92490577697754%
/usr/local/lib/python3.7/dist-packages/i
/usr/local/lib/python3.7/dist-packages/i

```

Figura 19: Desempenho (sMAPE): GRU

```

Evaluation Time: 0.013328000000001339
sMAPE: 26.836439967155457%
/usr/local/lib/python3.7/dist-packages/i
/usr/local/lib/python3.7/dist-packages/i

```

Figura 20: Desempenho (sMAPE): LSTM

Por fim, a Figura 21 mostra o gráfico dos dados originais (linha azul) e dos dados preditos (linha verde), no qual é possível observar que os valores obtidos se assemelham com o comportamento do *dataset* real, onde o gráfico da parte superior pertence ao resultado do modelo GRU e o gráfico da parte inferior pertence ao resultado do modelo LSTM:

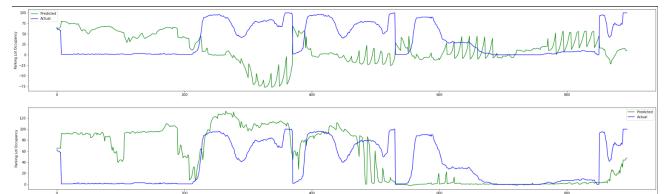


Figura 21: Gráfico GRU e LSTM PUC

C. Família Yolov5

Em relação a entradas que contemplam posições de câmera mais variados como, por exemplo, a partir de uma perspectiva horizontal, o treinamento do *dataset* Utmach no Yolov5 performa bem, exibindo uma acurácia mais generalista que o treinamento do mesmo modelo no *dataset* PKlot. Os resultados do treinamento com o Utmach podem ser visualizados a seguir:

Tabela XI: Resultados do treinamento do Utmach

Epoch	box_loss	obj_loss	cls_loss	precision	recall
Epoch 0	0.11097	0.18436	0.026413	0.12794	0.29431
Epoch 1	0.085018	0.20229	0.01702	0.28772	0.51597
Epoch 2	0.079301	0.18079	0.010748	0.44984	0.58118
Epoch 3	0.073811	0.17959	0.0079434	0.66861	0.62822
Epoch 4	0.070068	0.17357	0.0064685	0.49895	0.64923
Epoch 5	0.064696	0.17091	0.0054779	0.76307	0.70748
Epoch 6	0.06195	0.16248	0.0051046	0.88327	0.75848
Epoch 7	0.058122	0.1608	0.0047664	0.92867	0.75928
Epoch 8	0.056372	0.15807	0.0043836	0.94228	0.77398
Epoch 9	0.05484	0.16516	0.0044226	0.94137	0.7824

Em relação a entradas que contemplam uma posição de câmera fixa (relativo ao Centro de Informática), o treinamento do *dataset* Utmach no Yolov5 performa bem, exibindo uma

acurácia menos generalista, mas que performa bem para as imagens que respeitem as mesmas condições de captura de treino. Os resultados do treinamento com o *dataset* CIn podem ser visualizados a seguir:

Tabela XII: Resultados do treinamento do Utmach

Epoch	box_loss	obj_loss	cls_loss	precision	recall
Epoch 0	0.10668	0.13798	0.014541	0.70791	0.29431
Epoch 1	0.087411	0.12029	0.010858	0.39176	0.51597
Epoch 2	0.082539	0.11491	0.0084416	0.46814	0.58118
Epoch 3	0.075315	0.11211	0.51344	0.66861	0.62822
Epoch 4	0.069067	0.11075	0.0060282	0.49895	0.64923
Epoch 5	0.063843	0.10589	0.0053804	0.76307	0.70748
Epoch 6	0.060514	0.10649	0.0050291	0.88327	0.75848
Epoch 7	0.056627	0.098825	0.0045371	0.92867	0.75928
Epoch 8	0.053343	0.097605	0.0042553	0.94228	0.77398
Epoch 9	0.050398	0.09454	0.0040075	0.94137	0.7824

VII. CONCLUSÕES E DISCUSSÕES

Todos vídeos utilizados e gerados neste estudo podem ser encontrados em uma pasta no *Google Drive*¹. Além disso, os códigos podem ser encontrados no *GitHub*².

Ao observar os resultados da tarefa de detecção de ocupação do estacionamento, observamos que os modelos foram efetivos, precisando-se apenas pensar no tradeoff entre o tempo de treinamento contra a acurácia, uma vez que o AlexNet apresenta resultados melhores que o mAlexNet mas utilizando-se de um tempo de treinamento bem maior. Apesar disso, os autores concordam que treinar os modelos utilizando de bases de dados diversas que misturam condições climáticas e de luz reduz a acurácia do modelo, porém o torna generalista como observado na inferência no CIn. Ademais, os autores estão certos de que um treinamento dos modelos em um dataset capturado no CIn seria muito benéfico para uma melhor análise e especificação da solução, voltada para o CIn.

Quanto à previsão de ocupação, a predição realizada, de apenas alguns minutos à frente, não se mostra muito prática, mas os autores reconhecem que é um primeiro passo em direção a um preditor com um passo de tempo maior. Além disso, a alta acurácia das previsões (no estacionamento da PUC) mostra o grande potencial da solução.

REFERÊNCIAS

- [1] Paulo R.L. de Almeida, Luiz S. Oliveira, Alceu S. Britto, Eunelson J. Silva, Alessandro L. Koerich, PKLot – A robust dataset for parking lot classification, Expert Systems with Applications, Volume 42, Issue 11, 2015, Pages 4937-4949.
- [2] Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, Claudio Gennaro, Carlo Meghini, Claudio Vairo, Deep learning for decentralized parking lot occupancy detection, Expert Systems with Applications, Volume 72, 2017, Pages 327-334
- [3] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, Pages 779-788.
- [4] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv, 2014.
- [5] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation, Volume 9, Number 8, Pages 1735–1780, 1997.
- [6] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, ‘Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling’. arXiv, 2014.
- [7] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, CNR-PARK+EXT A dataset for visual occupancy detection of parking lots. [Online]. Available: <http://cnrpark.it/>. [Accessed: 06-Sep-2022].
- [8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ‘ImageNet Classification with Deep Convolutional Neural Networks’, Advances in Neural Information Processing Systems, 2012, Volume 25.
- [9] P. Mayur, YOLO V5 — Explained and Demystified, Towards AI Computer Vision, 2020. [Online]. Available: <https://towardsai.net/p/computer-vision/yolo-v5%E2%80%8A-%E2%80%8Aexplained-and-demystified>. [Accessed: 25-10-2022]
- [10] C. Wallner, M. Alam, M. Drysch, J. M. Wagner, A Highly Reliable Convolutional Neural Network Based Soft Tissue Sarcoma Metastasis Detection from Chest X-ray Images: A Retrospective Cohort Study, MDPI, Cancers, 2021, Volume 13, Number 4961. DOI: 10.3390/cancers13194961
- [11] R. Xu, H. Lin, K. Lu, L. Cao, Y. Liu, A Forest Fire Detection System Based on Ensemble Learning, MDPI, Forests, 2021, 12, 217. DOI: 10.3390/f12020217
- [12] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations(v6.2). Zenodo. <https://doi.org/10.5281/zenodo.7002879>. [Online]. Available: https://pytorch.org/hub/ultralytics_yolov5/. [Accessed: 25-10-2022]
- [13] Author: jcarrion5@utmachala.edu.ec, Type: Open Source Dataset, Journal: Roboflow Universe, Data: August - 2022, Available: https://universe.roboflow.com/jcarrion5-utmachala-edu-ec/utmach_fulldataset [visited on 2022-10-25]
- [14] Coco dataset available at: <https://cocodataset.org/home>

¹<https://drive.google.com/drive/folders/1tNaNYsbh1iKy5lgIHqIDZ6uO5ybweSOq>

²https://github.com/Gvascons/Parking_Spot_Detection