Relatório - Andamento da Rede Neural Geradora de Road Maps

Agosto - 2023

A rede de segmentação semântica escolhida foi a UNET, ela possui bons conteúdos de pesquisa online e não é uma rede pesada em comparação a outras redes de segmentação semântica existentes. O processo de tratamento de dados e treinamento da rede têm sido realizados no site *Kaggle*, que possui um ambiente para programação, e também que os processos sejam realizados nas máquinas virtuais. Entretanto, esse recurso é limitado, então fez-se necessário buscar técnicas e soluções para que as tarefas pudessem ser realizadas.

O problema enfrentado inicialmente durante esse mês para a realização do treinamento multiclasse da rede, foi o de tamanho de dados. Os dados de entrada tem em média 6GB no total, então ao serem carregados para ram ocupam esse espaço mais os espaços gastos com o processamento desses dados durante o treinamento e as variáveis criadas.

Para resolver isso, foi usado o 'Batch Generator'. Um método que permite que os dados sejam importados aos poucos, por lotes, e apenas o que será necessário usar naquela etapa, na próxima etapa, quando for necessário um novo lote, os dados são descartados e novos são importados. Esse método também resolveu a questão de longo tempo de processamento dos dados do ground truth, que devem passar por uma função chamada '*to_categorical*' que gasta muito processamento, tempo e também gera arquivos de saída muito grandes. Sendo assim cada lote pode ser processado na hora que for ser usado. Isso aumenta o tempo do treinamento, porém auxilia nos outros dois pontos sobre processamento e memória.

Para ser possível a utilização desse método, os arrays de dados que antes estavam salvos em formato .bin tiveram que ser reprocessados e salvos em arquivo formato CSV. Esse processo também foi realizado um método que processa o arquivo e salva em partes, pois realizar todo o processo e salvar 80.000 arrays em um CSV sequencialmente (40.000 do Input-X e 40.000 de Ground Truth-Y) demanda bastante memória e processamento.

Logo após isso, tornou-se possível realizar os treinamentos, porém verificou-se que o tempo gasto para o treinamento é extenso, sendo assim, não possível concluí-lo no kaggle, já que seu uso é limitado a 12 horas consecutivas de execução de processo do notebook, e 20 horas não consecutivas semanais de utilização da GPU fornecida.

Segue alguns valores, para que seja possível a compreensão sobre o tempo gasto:

- 7000 dados de treino e 3000 de teste
    - 1 epoch: 40 minutos para conclusão (batch tamanho 16)
- 40000 dados de treino e 10000 de teste
    - 1 epoch: 12 horas para conclusão (batch tamanho 64)

Foi pensado em trocar a rede por uma UNET otimizada com menos parâmetros, porém ao realizar testes, a UNET mostrou-se ocupar a mesma quantidade de espaço e contendo menos parâmetros que outras redes simplificadas que prometiam ser mais leves.

Exemplo:

- Squeeze UNET [1]
    - Parâmetros: 2.503.777
    - RAM: 2.3GB
    - GPU: 13.7 GB
- UNET [3]
    - Parâmetros: 1.941.089
    - RAM: 2.3GB
    - GPU: 13.7 GB

Sendo assim, decidiu seguir-se buscando outras técnicas para melhorar o treinamento e diminuir o tempo desse processo [2].
Foi implementado no treinamento o '*ModelCheckpoint'* e o '*EarlyStopping'*, o que permitiu que os dados e parâmetros de treinamento pudessem ser salvos mesmo se o processo fosse interrompido.
A utilização de um modelo pré-treinado tem sido estudado e testado no momento, para a otimização desse processo de treinamento e também visando um melhor resultado. Exemplos de modelos pré-treinados: ResNet [4], VGG16 [5], Inception V3 [6].
Uma tentativa de migração para o Google Collab foi iniciada na expectativa de se utilizar hardwares que permitam a execução do treinamento completo, porém ainda está sendo estudado para entender a sua lógica de funcionamento baseado em créditos gratuitos e pagos. Também estou negociando com professores do IFES para a utilização de um dos computadores disponíveis no laboratório de extensão da instituição que possui hardware que atende e liberdade de tempo de execução.
A documentação do processo e os arquivos que estão sendo utilizados do kaggle estão sendo salvos no repositório do GitHub (https://github.com/luddias/RoadMapper-UNET).

Referências

[1] Iontra, L. (2023). *Squeeze-unet Semantic Segmentation for embedded devices*. [online] GitHub.
Disponível em: https://github.com/lhelontra/squeeze-unet/tree/master

[2] Dias, L. (2023). *DeepLearning-About*. [online] GitHub.
Disponível em:https://github.com/luddias/DeepLearning-About

[3] Olaf Ronneberger, Fischer, P. and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv (Cornell University)*. doi:https://doi.org/10.48550/arxiv.1505.04597.

[4] He, K., Zhang, X., Ren, S. and Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv (Cornell University)*. doi:https://doi.org/10.48550/arxiv.1512.03385.

[5] Simonyan, K. and Zisserman, A. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. [online] arXiv.org. Available at: https://arxiv.org/abs/1409.1556.

[6] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2015). *Rethinking the Inception Architecture for Computer Vision*. [online] arXiv.org. Available at: https://arxiv.org/abs/1512.00567.