UFSC

UNIVERSIDADE FEDERAL DE SANTA CATARINA

TÉCNICAS DE PRUNING PARA COMPACTAÇÃO DE UM MODELO DE SEGMENTAÇÃO SEMÂNTICA DE DETECÇÃO DE CAMINHO NAVEGÁVEL

Gabriel Machado | Aldo von Wangenheim

Atualmente, há diversos estudos que utilizam a visão passiva como proposta para a detecção do caminho e obstáculos em veículos autônomos, todavia a maioria destes levam em consideração apenas estradas de países desenvolvidos [1]. Nesse sentido, há a necessidade do veículo conseguir lidar com a situação dos pavimentos de países em desenvolvimento e, ainda, com uma performance que transmita segurança em questões de tempo de processamento em plataformas embarcadas, as quais possuem recursos gráficos limitados.

MÉTODOS

A rede original de segmentação semântica para detecção do caminho foi treinada utilizando a arquitetura U-NET e também, como backbone, a resnet-34. Sendo o modelo treinado através da biblioteca fast.ai, este compatível com PyTorch, optou-se por realizar os procedimentos de *pruning* também com o PyTorch.

Em seguida, o objetivo é tornar esse modelo treinado leve e compacto, de forma a tornar possível sua instalação em uma plataforma embarcada dentro de um veículo, dada as limitações gráficas desses dispositivos. Sendo o veículo um elemento em movimento, faz-se necessário que o modelo consiga além de fornecer a detecção com boa acurácia, performar também de forma rápida, visto que envolverá a segurança de pessoas no trânsito. Nesse sentido, busca-se um *trade-off* entre acurácia e velocidade.

Para isso, uma técnica promissora é a de *pruning*, que nada mais é que remover conexões e/ou nós da rede treinada, conforme figura 1 [2], tornando-a menor e mais compacta e, por consequência, com uma inferência mais rápida.

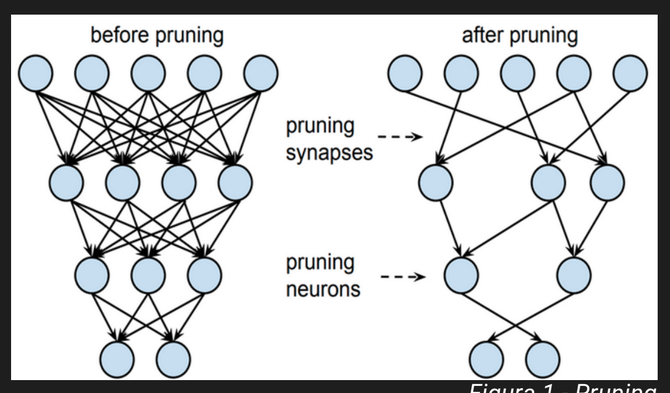

Figura 1 - Pruning

Entretanto, os nós e conexões não são removidos de forma aleatória, é necessário avaliar aqueles que possuem pouca influência no resultado final do modelo, ou seja, aqueles que são “descartáveis”. Há várias formas de articular esse procedimento, neste trabalho optamos por testar a **estruturada** (*structured pruning)* e a **não-estruturada** (*unstructured pruning)*. A primeira, consiste em desativar os pesos desnecessários zerando eles, ou seja, substituindo seus valores por zero, sem alterar a estrutura do modelo. Já a segunda, consiste em remover completamente os pesos, alterando assim a estrutura do modelo. Ambos os experimentos foram testados removendo os 25% pesos menores ranqueados.

RESULTADOS

Aplicando dois dos principais metódos de *pruning,* verificamos a eficiência e impactos que cada um gera em um modelo treinado. Na figura 2 é possível visualizar a comparação entre as saídas geradas pelo modelo original e pelo modelo após aplicado as técnicas de *pruning*. Em questões qualitativas, o modelo que foi aplicado o *unstructured pruning* manteve a sua acurácia intacta entre as classes que ele se propôs a classificar, enquanto que o *structured pruning* perdeu uma quantidade substancial de qualidade, reduzindo de 95,95% para 87,88%, apenas mantendo-se em patamares acima de 80% devido ao fato de manter a qualidade na classificação do *background*, que é a maior parte do frame. Já em relação ao tempo de inferência, conforme a tabela 1, houve uma redução de cerca de 5% para o *unstructured pruning* e 8% para o *structured pruning*.

Figura 2 - Resultados

Modelo	Tempo Gasto	FPS	Tamanho	Acurácia
Unstructured Pruning	0,096 s	10,42	157 Mb	95,96%
Structured Pruning	0,093 s	10,75	157 Mb	87,88%
Original	0,101 s	9,9	472 Mb	95,95%

Tabela 1 - Benchmarks

CONCLUSÕES

Com a utilização da técnica de *pruning*, foi possível verificar uma considerável melhora em questões de tempo de processamento na eficiência da rede. Tornou-se nítido que a melhor escolha para o caso em questão, é a opção com *unstructured pruning*, pois não comprometeu a qualidade do modelo. Ao aplicar o *strucutred pruning,* foram excluídas certas camadas convolucionais que eram responsáveis pela detecção de classes com menores pesos no balanceamento do *dataset* . Por esse motivo, atingiu uma qualidade razoável (87,88%), apesar de visualmente não desempenhar um bom trabalho. Com isso, o próximo passo é verificar como esse modelo compactado irá responder ao converte-lo para um framework ainda mais eficiente, como o TensorRT e, em seguida, embarca-lo em um dispositivo, como a NVIDIA Jetson TX2.

[1] Rateke, T., von Wangenheim, A. Road surface detection and differentiation considering surface damages. Auton Robot 45, 299–312 (2021). https://doi.org/10.1007/s10514-020-09964-3

[2] Bandaru, Rohit. Pruning Neural Networks. https://towardsdatascience.com/pruning-neural-networks-1bb3ab5791f9