推論ネットワーク

- 3D物体検出のベースネットワークは Super-Fast-Accurate-3D-Object-Detection-PyTorch[^1] (SFA3D) を選択した
  車両全方位の LiDAR データから BEV 鳥瞰画像を生成し、fpn_resnet_18 を用いた CenterNet で物体中心座標を推論する

- BEV 画像は 608 x 608 x 3 の ２次元画像であるが、R/G/Bに相当する画素データとして、intensity / height / density としている。

  1. intensity は LiDAR データの輝度である。輝度はsoftmax 関数で圧縮された最大1.0の値をとる。BEV１画素の中に複数の点が入った場合 height (z 軸) 最大の点の輝度で代表する。
  2. height は LiDAR データの z軸(物体の高さ)であるが、BEV１画素の中に複数の点が入った場合は最大値を保持する。
  3. density は、BEV１画素の中に入った点の数を log 圧縮した値である。

  このような RGB ２次元画像とした BEV 画像に CenterNet を適用することで、高い精度を得ている。

- CenterNet の出力 heatmap (152x152x3(class)) 画像はBEV画像の 1/4 解像度であり、画素の分解能が低い。 検出物体位置の分解能を上げるため、cen_offset (152x152x2(xy) heatmap １画素内の２次元オフセット値) を推論することで位置精度を出す構造である。

- 物体のサイズ、向きを同時に推論できるが、今回の課題では物体の中心座標のみ求められたので、処理の高速化のために heatmap と cen_offset のみに絞った。

課題に合わせた学習

- 学習には提供された OperaDataset のうち、全方位でアノテーションされているデータのみ用い、shuffle した後に train 90% val 10% に分けて学習した

- SFA3D の学習環境は、kitti dataset を前提としている。このため、OperaDataset の nuScenes format から kitti format に変換した。
  nuScenes devkit の export_kitty.py を今回提供されたデータに合わせて改造して用いた。
  （自車前方の定義、アノテーションクラスなどを変更した）

- 課題に合わせ、BEV のサイズ、自車前方の方角 を変更して学習した
  sfa/config/kitti_config.py, train_config.py 等を修正 :
  BEV画素サイズ → 320x320 / 448 x 448 / 608 x 608
  boundary 設定 → 前後左右に 50m / 42m
  自車前方の方角 → 0° / 45°

- 今回の課題では物体の中心座標のみ求めるので、動作速度の観点でネットワークの出力を heatmap と center offset に絞った
  また、fpn_resnet_18 (sfa/models/fpn_resnet.py) の中の ReLU を ReLU6 に変更したところ量子化後の精度が若干改善した

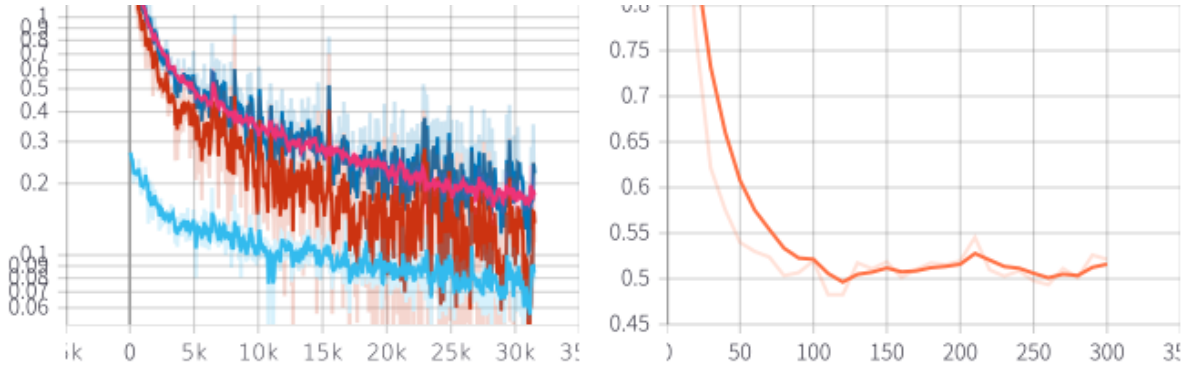


図1. 学習の経過 : training loss / validation loss

^1 SFA3D

pytorch → TFlite 変換 → int8 量子化

学習後の pytorch ネットワークを TFlite に変換し、8bit 量子化を行った

- pytorch model を openvino2tensorflow[^2] で TFlite に変換し、Training 後の量子化を行って、8bit に量子化した。
  Training 後の量子化で用いるネットワークの入力画像に training/validation で用いた BEV 画像を用いることで、量子化後の精度が向上した。
- TFlite のネットワークを編集し、Pad+Conv2D(valid) を Conv2D(same) に置き換えた(図2)
  pytorch → TFlite 変換で、Conv2D の仕様の不整合[^3]から pytorch:Conv2D(same) -> TFlite:Pad+Conv2D(valid) に変換されてしまう(余分な Pad 処理が発生する)
  Conv2D の stride 設定が 1 のときは単純に Pad をバイパスし、valid を same に書き換えればよいが、stride が 2 のときに Conv2D の出力位相が合わなくなる。strideが 2 のノードは FPGA のアクセラレータの設定を修正して位相を合わせるようにした

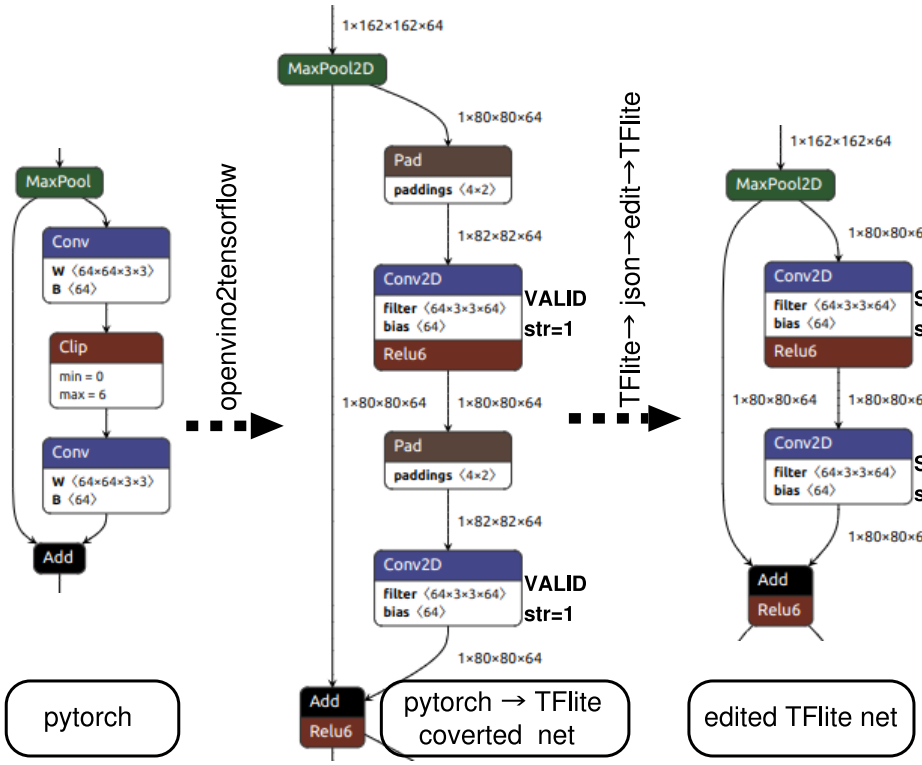


図2. pytorch → TFlite 変換で余分なPADが発生する問題と対処

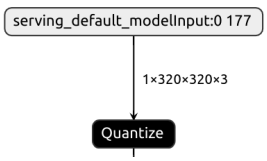
[^2] openvino2tensorflow

[^3] comparing-conv2d-with-padding-between-tensorflow-and-pytorch

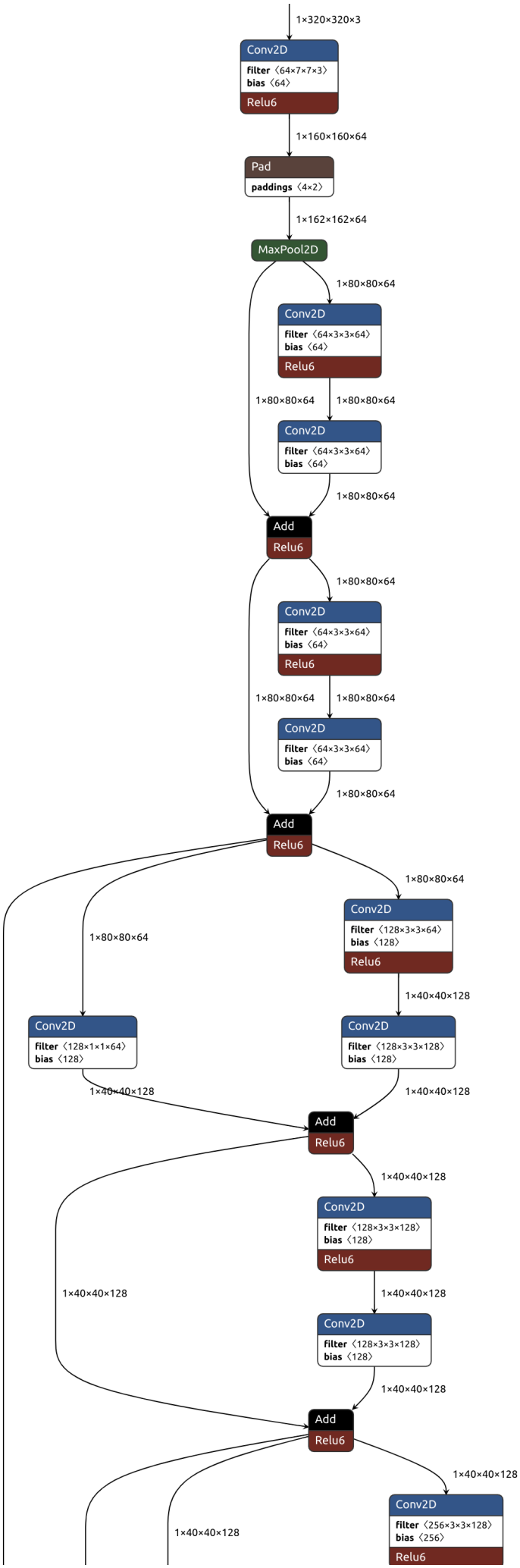
TFlite ネットワークの編集

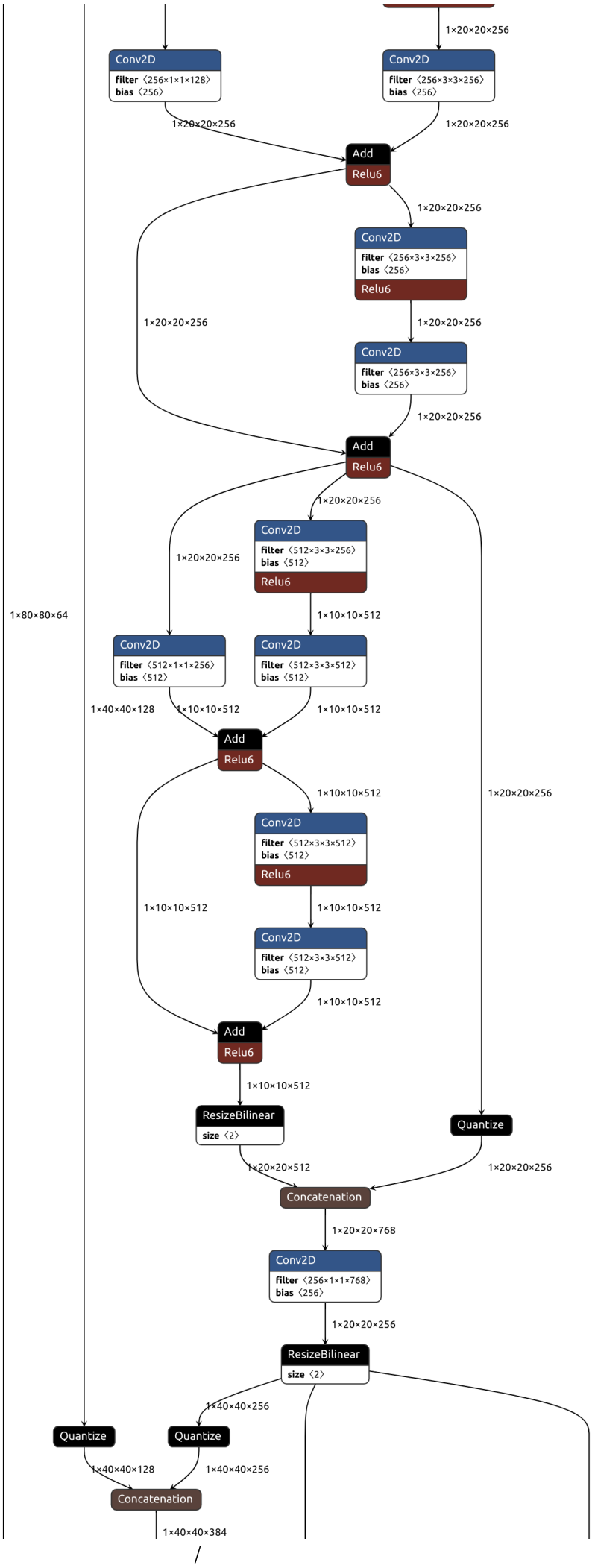
TFlite のモデルを一旦 json に変換し、json を python script で編集し、再び TFlite に戻す。
tflite2json2tflite

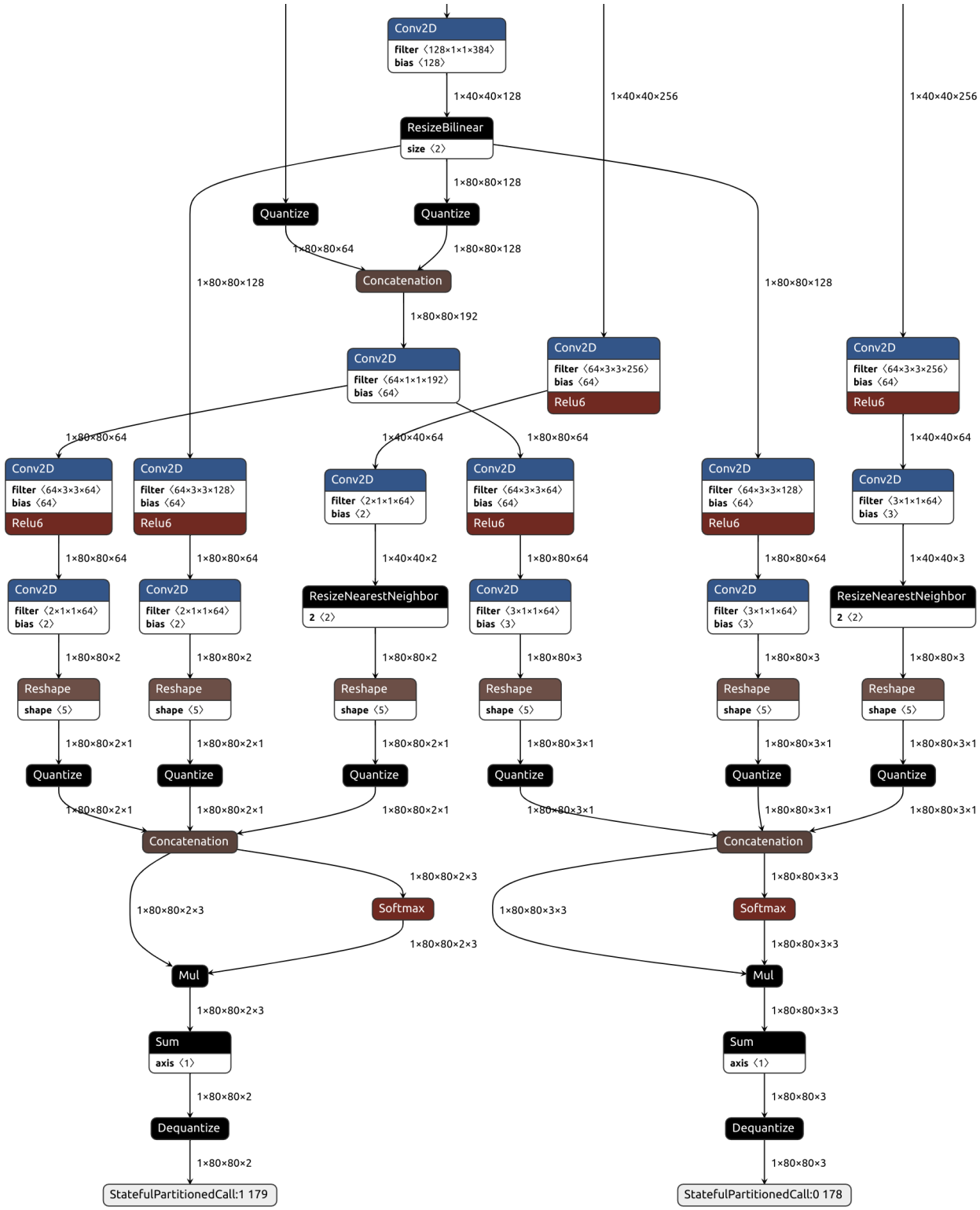
??? "TFlite model graph (320x320)"



/


1×320×320×3
Conv2D
filter 〈64×7×7×3〉
bias 〈64〉
Relu6
1×160×160×64
Pad
paddings 〈4×2〉
1×162×162×64
MaxPool2D
1×80×80×64
Conv2D
filter 〈64×3×3×64〉
bias 〈64〉
Relu6
1×80×80×64
1×80×80×64
Conv2D
filter 〈64×3×3×64〉
bias 〈64〉
1×80×80×64
Add
Relu6
1×80×80×64
Conv2D
filter 〈64×3×3×64〉
bias 〈64〉
Relu6
1×80×80×64
1×80×80×64
Conv2D
filter 〈64×3×3×64〉
bias 〈64〉
1×80×80×64
Add
Relu6
1×80×80×64
1×80×80×64
Conv2D
filter 〈128×3×3×64〉
bias 〈128〉
Relu6
1×40×40×128
Conv2D
filter 〈128×1×1×64〉
bias 〈128〉
1×40×40×128
Conv2D
filter 〈128×3×3×128〉
bias 〈128〉
1×40×40×128
Add
Relu6
1×40×40×128
Conv2D
filter 〈128×3×3×128〉
bias 〈128〉
Relu6
1×40×40×128
1×40×40×128
Conv2D
filter 〈128×3×3×128〉
bias 〈128〉
1×40×40×128
Add
Relu6
1×40×40×128
1×40×40×128
Conv2D
filter 〈256×3×3×128〉
bias 〈256〉
Relu6


1×20×20×256
Conv2D
filter 〈256×1×1×128〉
bias 〈256〉
Conv2D
filter 〈256×3×3×256〉
bias 〈256〉
1×20×20×256
1×20×20×256
Add
Relu6
1×20×20×256
Conv2D
filter 〈256×3×3×256〉
bias 〈256〉
Relu6
1×20×20×256
1×20×20×256
Conv2D
filter 〈256×3×3×256〉
bias 〈256〉
1×20×20×256
Add
Relu6
1×20×20×256
1×20×20×256
Conv2D
filter 〈512×3×3×256〉
bias 〈512〉
Relu6
1×80×80×64
1×10×10×512
Conv2D
filter 〈512×1×1×256〉
bias 〈512〉
Conv2D
filter 〈512×3×3×512〉
bias 〈512〉
1×40×40×128
1×10×10×512
1×10×10×512
Add
Relu6
1×10×10×512
1×20×20×256
Conv2D
filter 〈512×3×3×512〉
bias 〈512〉
Relu6
1×10×10×512
1×10×10×512
Conv2D
filter 〈512×3×3×512〉
bias 〈512〉
1×10×10×512
Add
Relu6
1×10×10×512
ResizeBilinear
size 〈2〉
Quantize
1×20×20×512
1×20×20×256
Concatenation
1×20×20×768
Conv2D
filter 〈256×1×1×768〉
bias 〈256〉
1×20×20×256
ResizeBilinear
size 〈2〉
1×40×40×256
Quantize
Quantize
1×40×40×128
1×40×40×256
Concatenation
1×40×40×384


Conv2D
filter 〈128×1×1×384〉
bias 〈128〉
1×40×40×128
1×40×40×256
1×40×40×256
ResizeBilinear
size 〈2〉
1×80×80×128
Quantize
Quantize
1×80×80×64
1×80×80×128
1×80×80×128
1×80×80×128
Concatenation
1×80×80×192
Conv2D
filter 〈64×1×1×192〉
bias 〈64〉
Conv2D
filter 〈64×3×3×256〉
bias 〈64〉
Relu6
Conv2D
filter 〈64×3×3×256〉
bias 〈64〉
Relu6
1×80×80×64
1×40×40×64
1×80×80×64
1×40×40×64
Conv2D
filter 〈64×3×3×64〉
bias 〈64〉
Relu6
Conv2D
filter 〈64×3×3×128〉
bias 〈64〉
Relu6
Conv2D
filter 〈2×1×1×64〉
bias 〈2〉
Conv2D
filter 〈64×3×3×64〉
bias 〈64〉
Relu6
Conv2D
filter 〈64×3×3×128〉
bias 〈64〉
Relu6
Conv2D
filter 〈3×1×1×64〉
bias 〈3〉
1×80×80×64
1×80×80×64
1×40×40×2
1×80×80×64
1×80×80×64
1×40×40×3
Conv2D
filter 〈2×1×1×64〉
bias 〈2〉
Conv2D
filter 〈2×1×1×64〉
bias 〈2〉
ResizeNearestNeighbor
2 〈2〉
Conv2D
filter 〈3×1×1×64〉
bias 〈3〉
Conv2D
filter 〈3×1×1×64〉
bias 〈3〉
ResizeNearestNeighbor
2 〈2〉
1×80×80×2
1×80×80×2
1×80×80×2
1×80×80×3
1×80×80×3
1×80×80×3
Reshape
shape 〈5〉
Reshape
shape 〈5〉
Reshape
shape 〈5〉
Reshape
shape 〈5〉
Reshape
shape 〈5〉
Reshape
shape 〈5〉
1×80×80×2×1
1×80×80×2×1
1×80×80×2×1
1×80×80×3×1
1×80×80×3×1
1×80×80×3×1
Quantize
Quantize
Quantize
Quantize
Quantize
Quantize
1×80×80×2×1
1×80×80×2×1
1×80×80×2×1
1×80×80×3×1
1×80×80×3×1
1×80×80×3×1
Concatenation
Concatenation
1×80×80×2×3
1×80×80×3×3
1×80×80×2×3
Softmax
1×80×80×3×3
Softmax
1×80×80×2×3
1×80×80×3×3
Mul
Mul
1×80×80×2×3
1×80×80×3×3
Sum
axis 〈1〉
Sum
axis 〈1〉
1×80×80×2
1×80×80×3
Dequantize
Dequantize
1×80×80×2
1×80×80×3
StatefulPartitionedCall:1 179
StatefulPartitionedCall:0 178