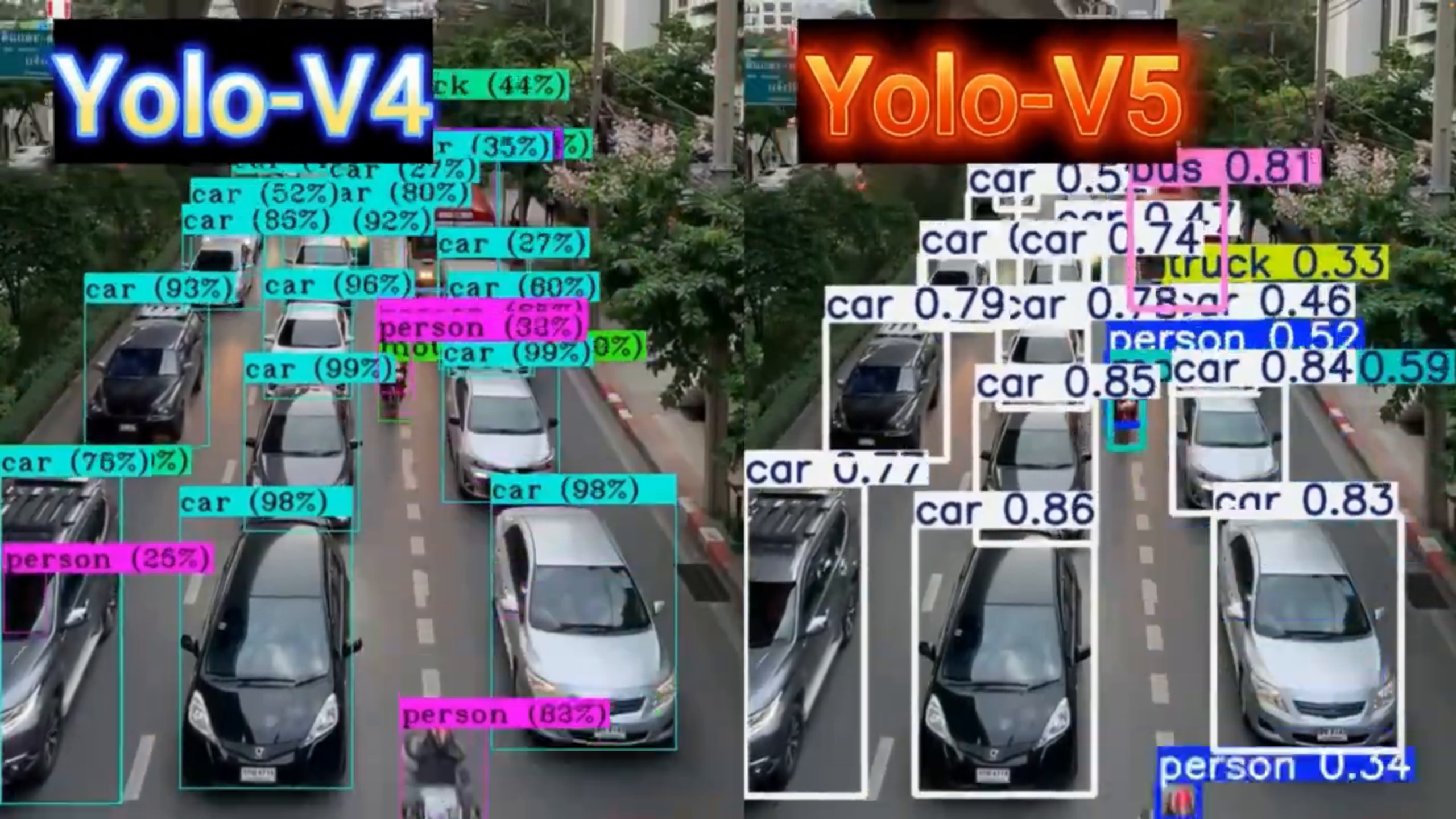
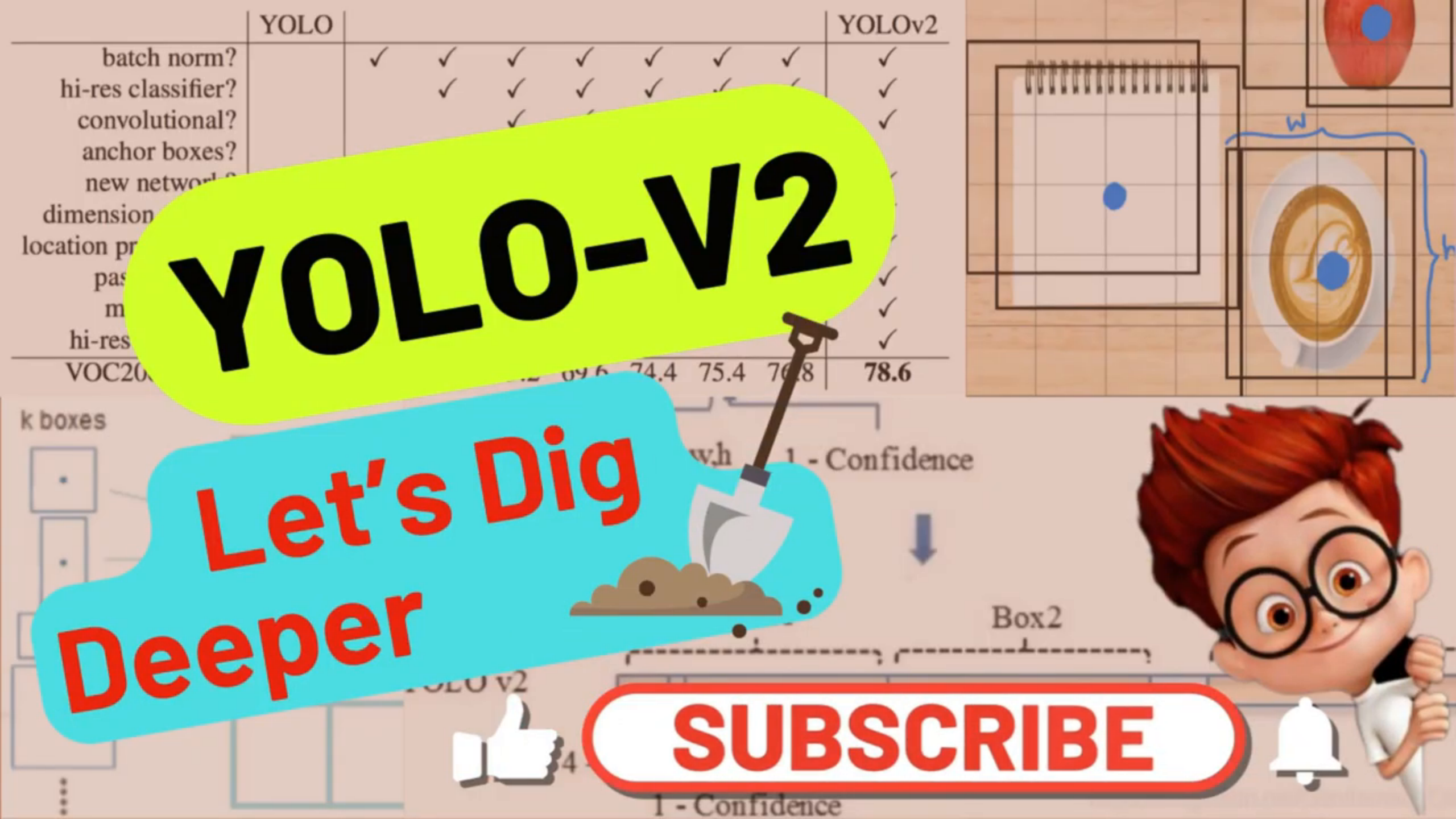

Yolo-V4
Yolo-V5
car (52%)
car (85%)
car (27%)
car (93%)
car (96%)
car (60%)
person (33%)
car (99%)
car (99%)
car (76%)
car (98%)
car (98%)
person (26%)
person (83%)
bus 0.81
car 0.74
truck 0.33
car 0.79
car 0.46
person 0.52
car 0.84
0.59
car 0.85
car 0.86
car 0.83
person 0.34


YOLO
YOLOv2
batch norm?
hi-res classifier?
convolutional?
anchor boxes?
78.6
YOLO-V2
Let's Dig Deeper
k boxes
1 - Confidence
Box2
SUBSCRIBE

You Only Look Once:
Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [13]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only

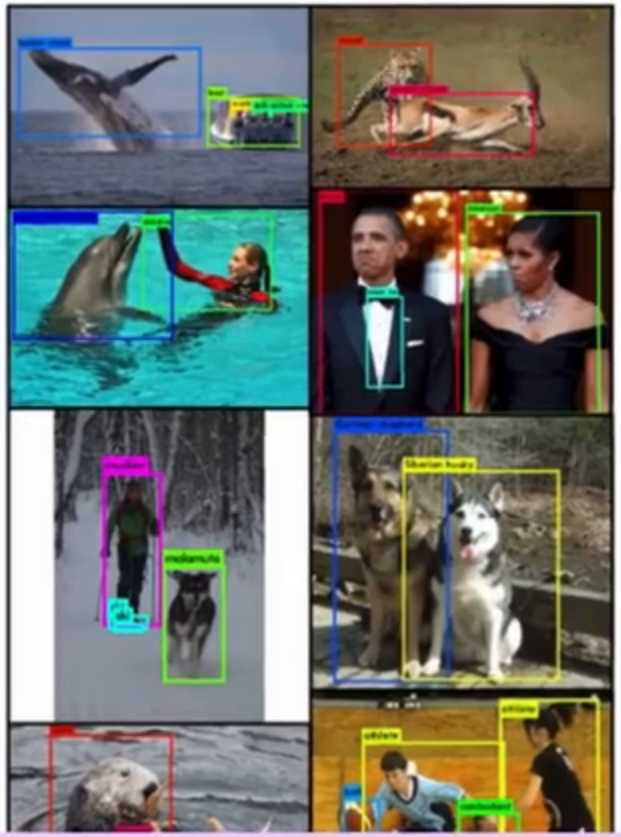
arXiv:1506.02640v5 [cs.CV] 9 May 2016

YOLO9000:
Better, Faster, Stronger

Joseph Redmon*†, Ali Farhadi*†
University of Washington*, Allen Institute for AI†
http://pjreddie.com/yolo9000/

Abstract

We introduce YOLO9000, a state-of-the-art, real-time object detection system that can detect over 9000 object categories. First we propose various improvements to the YOLO detection method, both novel and drawn from prior work. The improved model, YOLOv2, is state-of-the-art on standard detection tasks like PASCAL *VOC and COCO. Using a novel, multi-scale training method the same YOLOv2 model can run at varying sizes, offering an easy tradeoff between speed and accuracy. At 67 FPS, YOLOv2 gets 76.8 mAP on VOC 2007. At 40 FPS, YOLOv2 gets 78.6 mAP, outperforming state-of-the-art methods like Faster R-CNN with ResNet and SSD while still running significantly faster. Finally we propose a method to jointly train on object detection and classification. Using this method we train YOLO9000 simultaneously on the COCO detection dataset and the ImageNet classification dataset. Our joint training allows YOLO9000 to predict detections for object classes that don't have labelled detection data. We validate our approach on the ImageNet detection task. YOLO9000 gets 19.7 mAP on the ImageNet detection validation set despite only having detection data for 44 of the 200 classes. On the 156 classes not in COCO, YOLO9000 gets 16.0 mAP. But YOLO can detect more than just 200 classes; it predicts detections for more than 9000 different object categories. And it still runs in real-time.*



You Only Look Once:
Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [13]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only

arXiv:1506.02640v5 [cs.CV] 9 May 2016

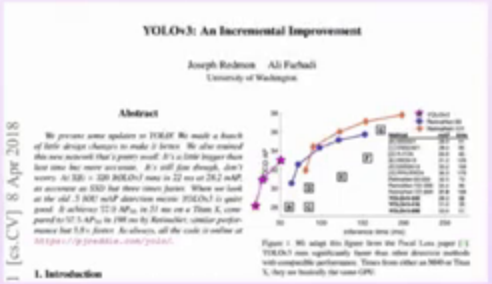
YOLO9000:
Better, Faster, Stronger

Joseph Redmon*†, Ali Farhadi*†
University of Washington*, Allen Institute for AI†
http://pjreddie.com/yolo9000/

Abstract

We introduce YOLO9000, a state-of-the-art, real-time object detection system that can detect over 9000 object categories. First we propose various improvements to the YOLO detection method, both novel and drawn from prior work. The improved model, YOLOv2, is state-of-the-art on standard detection tasks like PASCAL VOC and COCO. Using a novel, multi-scale training method the same YOLOv2 model can run at varying sizes, offering an easy tradeoff between speed and accuracy. At 67 FPS, YOLOv2 gets 76.8 mAP on VOC 2007. At 40 FPS, YOLOv2 gets 78.6 mAP, outperforming state-of-the-art methods like Faster R-CNN with ResNet and SSD while still running significantly faster. Finally we propose a method to jointly train on object detection and classification. Using this method we train YOLO9000 simultaneously on the COCO detection dataset and the ImageNet classification dataset. Our joint training allows YOLO9000 to predict detections for object classes that don't have labelled detection data. We validate our approach on the ImageNet detection task. YOLO9000 gets 19.7 mAP on the ImageNet detection validation set despite only having detection data for 44 of the 200 classes. On the 156 classes not in COCO, YOLO9000 gets 16.0 mAP. But YOLO can detect more than just 200 classes; it predicts detections for more than 9000 different object categories. And it still runs in real-time.

arXiv:1612.08242v1 [cs.CV] 25 Dec 2016



YOLOv3: An Incremental Improvement

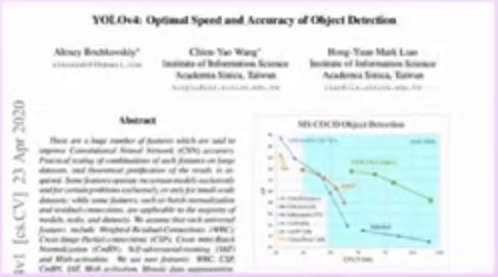
Joseph Redmon Ali Farhadi
University of Washington

Abstract

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP_{50} in 51 ms on a Titan X, compared to 57.5 AP_{50} in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at https://pjreddie.com/yolo/.

1. Introduction

arXiv:1804.02767v1 [cs.CV] 8 Apr 2018



YOLOv4: Optimal Speed and Accuracy of Object Detection

Alexey Bochkovskiy*
alexeyab84@gmail.com

Chien-Yao Wang*
Institute of Information Science
Academia Sinica, Taiwan
kinyiu@iis.sinica.edu.tw

Hong-Yuan Mark Liao
Institute of Information Science
Academia Sinica, Taiwan
liao@iis.sinica.edu.tw

Abstract

There are a huge number of features which are said to improve Convolutional Neural Network (CNN) accuracy. Practical testing of combinations of such features on large datasets, and theoretical justification of the result is required. Some features operate on certain models exclusively and for certain problems exclusively, or only for small-scale datasets; while some features, such as batch-normalization and residual-connections, are applicable to the majority of models, tasks, and datasets. We assume that such universal features include Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation. We use new features: WRC, CSP, CmBN, SAT, Mish activation, Mosaic data augmentation

arXiv:2004.10934v1 [cs.CV] 23 Apr 2020

YOLO-V5

You Only Look Once:
Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [13]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

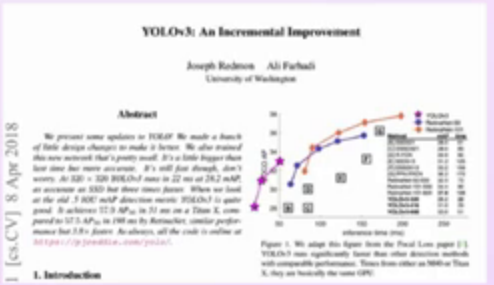
We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only

YOLO9000:
Better, Faster, Stronger

Joseph Redmon*†, Ali Farhadi*†
University of Washington*, Allen Institute for AI†
http://pjreddie.com/yolo9000/

Abstract

We introduce YOLO9000, a state-of-the-art, real-time object detection system that can detect over 9000 object categories. First we propose various improvements to the YOLO detection method, both novel and drawn from prior work. The improved model, YOLOv2, is state-of-the-art on standard detection tasks like PASCAL VOC and COCO. Using a novel, multi-scale training method the same YOLOv2 model can run at varying sizes, offering an easy tradeoff between speed and accuracy. At 67 FPS, YOLOv2 gets 76.8 mAP on VOC 2007. At 40 FPS, YOLOv2 gets 78.6 mAP, outperforming state-of-the-art methods like Faster R-CNN with ResNet and SSD while still running significantly faster. Finally we propose a method to jointly train on object detection and classification. Using this method we train YOLO9000 simultaneously on the COCO detection dataset and the ImageNet classification dataset. Our joint training allows YOLO9000 to predict detections for object classes that don't have labelled detection data. We validate our approach on the ImageNet detection task. YOLO9000 gets 19.7 mAP on the ImageNet detection validation set despite only having detection data for 44 of the 200 classes. On the 156 classes not in COCO, YOLO9000 gets 16.0 mAP. But YOLO can detect more than just 200 classes; it predicts detections for more than 9000 different object categories. And it still runs in real-time.



YOLOv3: An Incremental Improvement

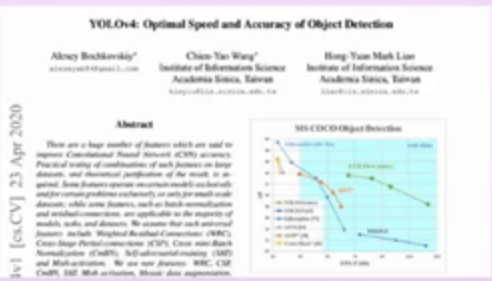
Joseph Redmon Ali Farhadi
University of Washington

Abstract

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP_{50} in 51 ms on a Titan X, compared to 57.5 AP_{50} in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at https://pjreddie.com/yolo/.

1. Introduction

Figure 1. We adapt this figure from the Focal Loss paper [9]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.



YOLOv4: Optimal Speed and Accuracy of Object Detection

Alexey Bochkovskiy*
alexeyab84@gmail.com

Chien-Yao Wang*
Institute of Information Science
Academia Sinica, Taiwan
kinyiu@iis.sinica.edu.tw

Hong-Yuan Mark Liao
Institute of Information Science
Academia Sinica, Taiwan
liao@iis.sinica.edu.tw

Abstract

There are a huge number of features which are said to improve Convolutional Neural Network (CNN) accuracy. Practical testing of combinations of such features on large datasets, and theoretical justification of the result, is required. Some features operate on certain models exclusively and for certain problems exclusively, or only for small-scale datasets; while some features, such as batch-normalization and residual-connections, are applicable to the majority of models, tasks, and datasets. We assume that such universal features include Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation. We use new features: WRC, CSP, CmBN, SAT, Mish activation, Mosaic data augmentation,

YOLO-V5

r/MachineLearning • 5 yr. ago
aloser

[News] YOLOv5 is Here: State-of-the-Art Object Detection at 140 FPS

News

YOLOv5 (PyTorch) was released by Ultralytics last night; early results show it runs inference extremely fast, weights can be exported to mobile, and it achieves state of the art on COCO.

It's insane how quickly SOTA for object detection is advancing. EfficientDet was just released in March. YOLOv4 in April. And now YOLOv5 in June.

- Ultralytics YOLOv5 Repo
- Writeup: YOLOv5 is Here: State-of-the-Art Object Detection at 140 FPS
- Tutorial: Training YOLOv5 on a Custom Dataset
- YOLOv5 Colab Notebook

YOLOv4: Optimal Speed and Accuracy of Object Detection

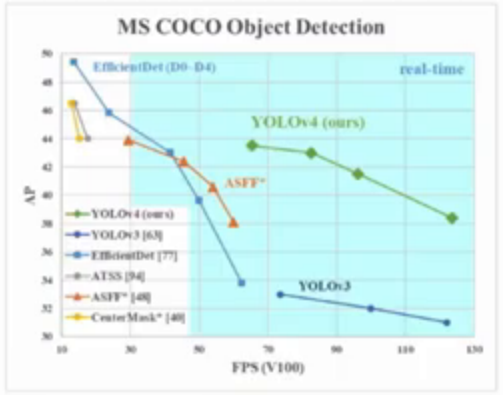
Alexey Bochkovskiy*
alexeyab84@gmail.com

Chien-Yao Wang*
Institute of Information Science
Academia Sinica, Taiwan
kinyiu@iis.sinica.edu.tw

Hong-Yuan Mark Liao
Institute of Information Science
Academia Sinica, Taiwan
liao@iis.sinica.edu.tw

Abstract

There are a huge number of features which are said to improve Convolutional Neural Network (CNN) accuracy. Practical testing of combinations of such features on large datasets, and theoretical justification of the result, is required. Some features operate on certain models exclusively and for certain problems exclusively, or only for small-scale datasets; while some features, such as batch-normalization and residual-connections, are applicable to the majority of models, tasks, and datasets. We assume that such universal features include Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation. We use new features: WRC, CSP, CmBN, SAT, Mish activation, Mosaic data augmentation,



README | Code of conduct | License | Security

YOLOv5
你只看一次v5

This repository represents Ultralytics open-source research into future object detection methods, and incorporates our lessons learned and best practices evolved over training thousands of models on custom client datasets with our previous YOLO repository https://github.com/ultralytics/yolov3. **All code and models are under active development, and are subject to modification or deletion without notice.** Use at your own risk.

- **June 22, 2020**: PANet updates: increased layers, reduced parameters, faster inference and improved mAP 364fcfd.
- **June 19, 2020**: FP16 as new default for smaller checkpoints and faster inference d4c6674.
- **June 9, 2020**: CSP updates: improved speed, size, and accuracy. Credit to @WongKinYiu for excellent CSP work.
- **May 27, 2020**: Public release of repo. YOLOv5 models are SOTA among all known YOLO implementations.
- **April 1, 2020**: Start development of future YOLOv3/YOLOv4-based PyTorch models in a range of compound-scaled sizes.

YOLOv4: Optimal Speed and Accuracy of Object Detection

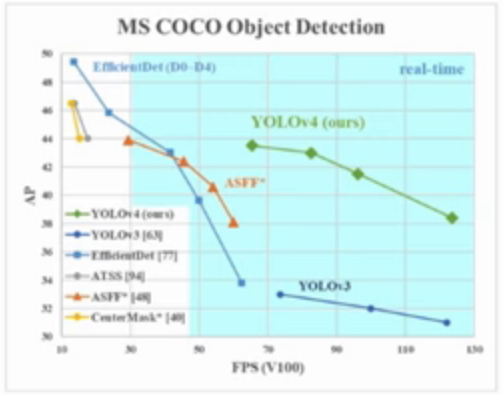
Alexey Bochkovskiy*
alexeyab84@gmail.com

Chien-Yao Wang*
Institute of Information Science
Academia Sinica, Taiwan
kinyiu@iis.sinica.edu.tw

Hong-Yuan Mark Liao
Institute of Information Science
Academia Sinica, Taiwan
liao@iis.sinica.edu.tw

Abstract

There are a huge number of features which are said to improve Convolutional Neural Network (CNN) accuracy. Practical testing of combinations of such features on large datasets, and theoretical justification of the result, is required. Some features operate on certain models exclusively and for certain problems exclusively, or only for small-scale datasets; while some features, such as batch-normalization and residual-connections, are applicable to the majority of models, tasks, and datasets. We assume that such universal features include Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation. We use new features: WRC, CSP, CmBN, SAT, Mish activation, Mosaic data augmentation,



README | Code of conduct | License | Security

YOLOv5
你只看一次v5

This repository represents Ultralytics open-source research into future object detection methods, and incorporates our lessons learned and best practices evolved over training thousands of models on custom client datasets with our previous YOLO repository https://github.com/ultralytics/yolov3. **All code and models are under active development, and are subject to modification or deletion without notice.** Use at your own risk.

- **June 22, 2020**: PANet updates: increased layers, reduced parameters, faster inference and improved mAP 364fcfd.
- **June 19, 2020**: FP16 as new default for smaller checkpoints and faster inference d4c6674.
- **June 9, 2020**: CSP updates: improved speed, size, and accuracy. Credit to @WongKinYiu for excellent CSP work.
- **May 27, 2020**: Public release of repo. YOLOv5 models are SOTA among all known YOLO implementations.
- **April 1, 2020**: Start development of future YOLOv3/YOLOv4-based PyTorch models in a range of compound-scaled sizes.

SOTA Claims

Second, YOLOv5 is fast – blazingly fast. In a YOLOv5 Colab notebook, running a Tesla P100, we saw inference times up to 0.007 seconds per image, meaning **140 frames per second (FPS)**! By contrast, YOLOv4 achieved 50 FPS after having been converted to the same Ultralytics PyTorch library.

Third, YOLOv5 is accurate. In our tests on the blood cell count and detection (BCCD) dataset, we achieved roughly 0.895 mean average precision (mAP) after training for just 100 epochs. Admittedly, we saw comparable performance from EfficientDet and YOLOv4, but it is rare to see such across-the-board performance improvements without any loss in accuracy.

Fourth, YOLOv5 is small. Specifically, a weights file for YOLOv5 is 27 megabytes. Our weights file for YOLOv4 (with Darknet architecture) is 244 megabytes. **YOLOv5 is nearly 90 percent smaller** than YOLOv4. This means YOLOv5 can be deployed to embedded devices much more easily.

SOTA Claims

Second, YOLOv5 is fast – blazingly fast. In a YOLOv5 Colab notebook, running a Tesla P100, we saw inference times up to 0.007 seconds per image, meaning **140 frames per second (FPS)**! By contrast, YOLOv4 achieved 50 FPS after having been converted to the same Ultralytics PyTorch library.

Third, YOLOv5 is accurate. In our tests on the blood cell count and detection (BCCD) dataset, we achieved roughly 0.895 mean average precision (mAP) after training for just 100 epochs. Admittedly, we saw comparable performance from EfficientDet and YOLOv4, but it is rare to see such across-the-board performance improvements without any loss in accuracy.

Fourth, YOLOv5 is small. Specifically, a weights file for YOLOv5 is 27 megabytes. Our weights file for YOLOv4 (with Darknet architecture) is 244 megabytes. **YOLOv5 is nearly 90 percent smaller** than YOLOv4. This means YOLOv5 can be deployed to embedded devices much more easily.

SOTA Claims

Second, YOLOv5 is fast – blazingly fast. In a YOLOv5 Colab notebook, running a Tesla P100, we saw inference times up to 0.007 seconds per image, meaning **140 frames per second (FPS)**! By contrast, YOLOv4 achieved 50 FPS after having been converted to the same Ultralytics PyTorch library.

Third, YOLOv5 is accurate. In our tests on the blood cell count and detection (BCCD) dataset, we achieved roughly 0.895 mean average precision (mAP) after training for just 100 epochs. Admittedly, we saw comparable performance from EfficientDet and YOLOv4, but it is rare to see such across-the-board performance improvements without any loss in accuracy.

Fourth, YOLOv5 is small. Specifically, a weights file for YOLOv5 is 27 megabytes. Our weights file for YOLOv4 (with Darknet architecture) is 244 megabytes. **YOLOv5 is nearly 90 percent smaller** than YOLOv4. This means YOLOv5 can be deployed to embedded devices much more easily.

SOTA Claims

Second, YOLOv5 is fast – blazingly fast. In a YOLOv5 Colab notebook, running a Tesla P100, we saw inference times up to 0.007 seconds per image, meaning **140 frames per second (FPS)**! By contrast, YOLOv4 achieved 50 FPS after having been converted to the same Ultralytics PyTorch library.

Third, YOLOv5 is accurate. In our tests on the blood cell count and detection (BCCD) dataset, we achieved roughly 0.895 mean average precision (mAP) after training for just 100 epochs. Admittedly, we saw comparable performance from EfficientDet and YOLOv4, but it is rare to see such across-the-board performance improvements without any loss in accuracy.

Fourth, YOLOv5 is small. Specifically, a weights file for YOLOv5 is 27 megabytes. Our weights file for YOLOv4 (with Darknet architecture) is 244 megabytes. **YOLOv5 is nearly 90 percent smaller** than YOLOv4. This means YOLOv5 can be deployed to embedded devices much more easily.

https://github.com/ultralytics/yolov5

AlexeyAB on Jun 11, 2020

Some notes on comparison: https://github.com/ultralytics/yolov5

- The latency shouldn't be measured with batch=32. The latency must be measured with batch=1, because the higher batch higher latency. The latency is the time of a complete data processing cycle, it cannot be less than processing a whole batch can take up to 1 second depends on batch-size

- If there is used batch=32 for both Yolov5 vs EfficientDet (I don't know), then this is ok, but only for Yolov5 vs EfficientDet, ar FPS (not for latency), it can't be compared with any other results where is batch=1

- Size of weights: yolov5x.pt - `366 MB` , yolov5s.pt - `27 MB`

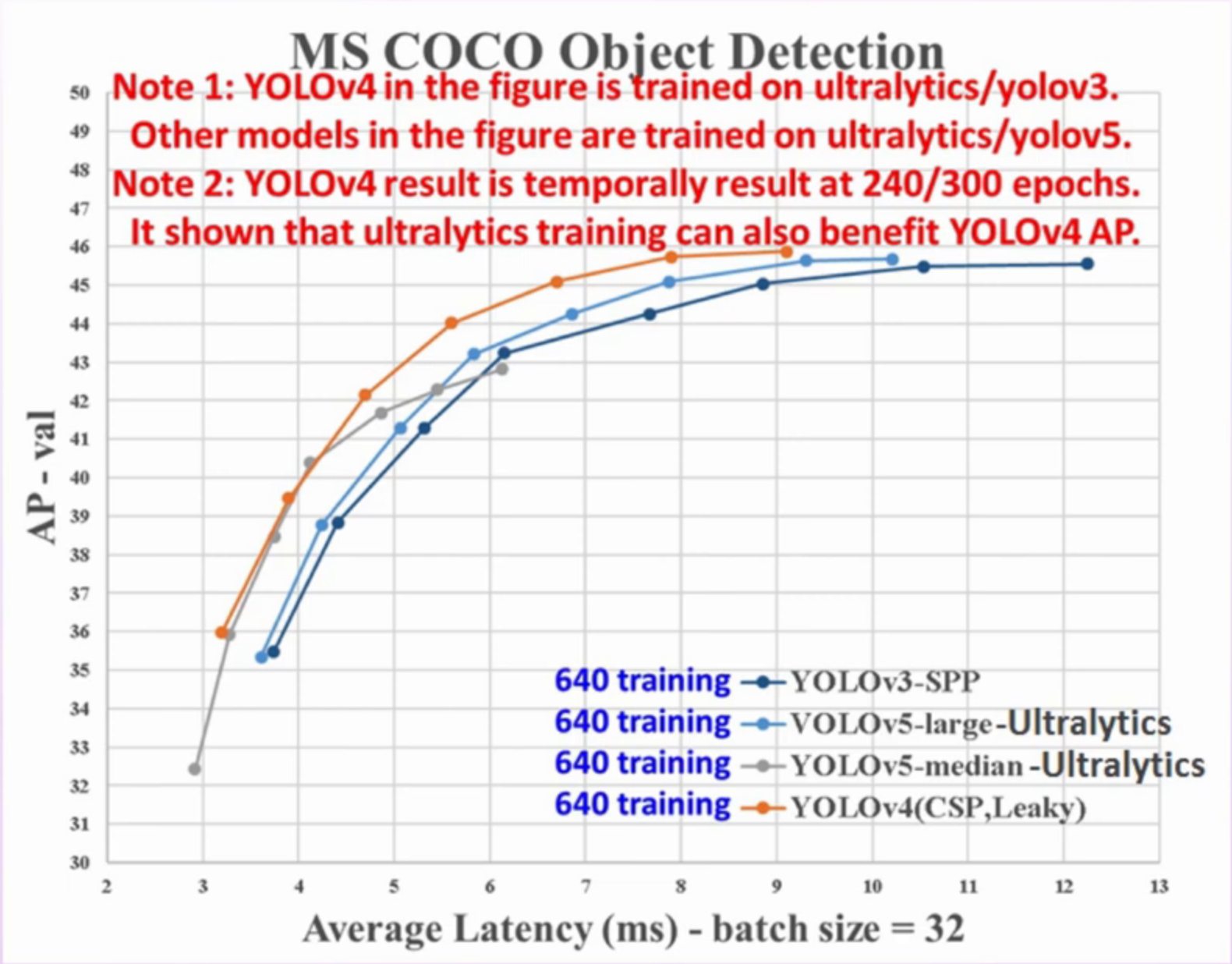
AlexeyAB on Jun 11, 2020

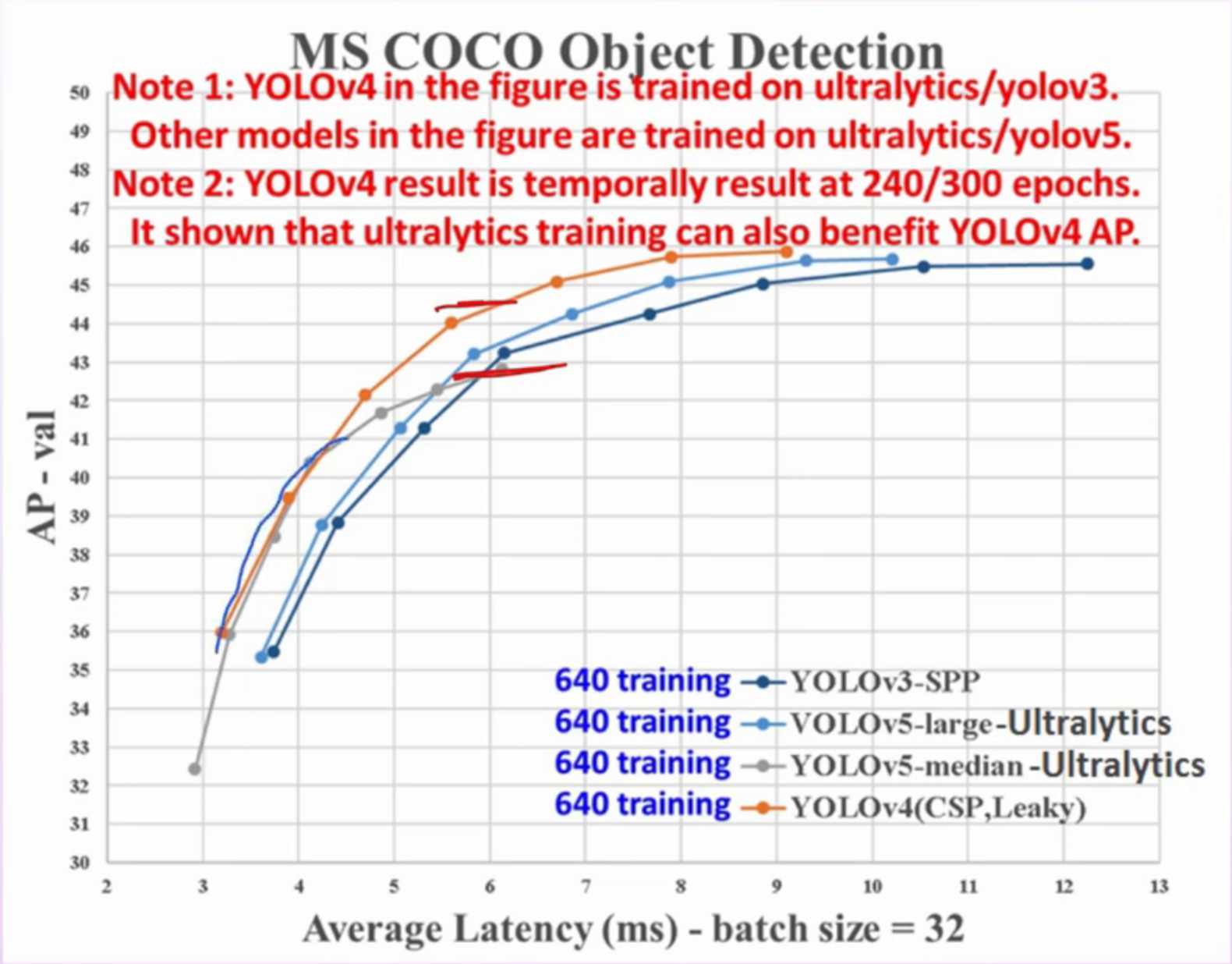
Some notes on comparison: https://github.com/ultralytics/yolov5

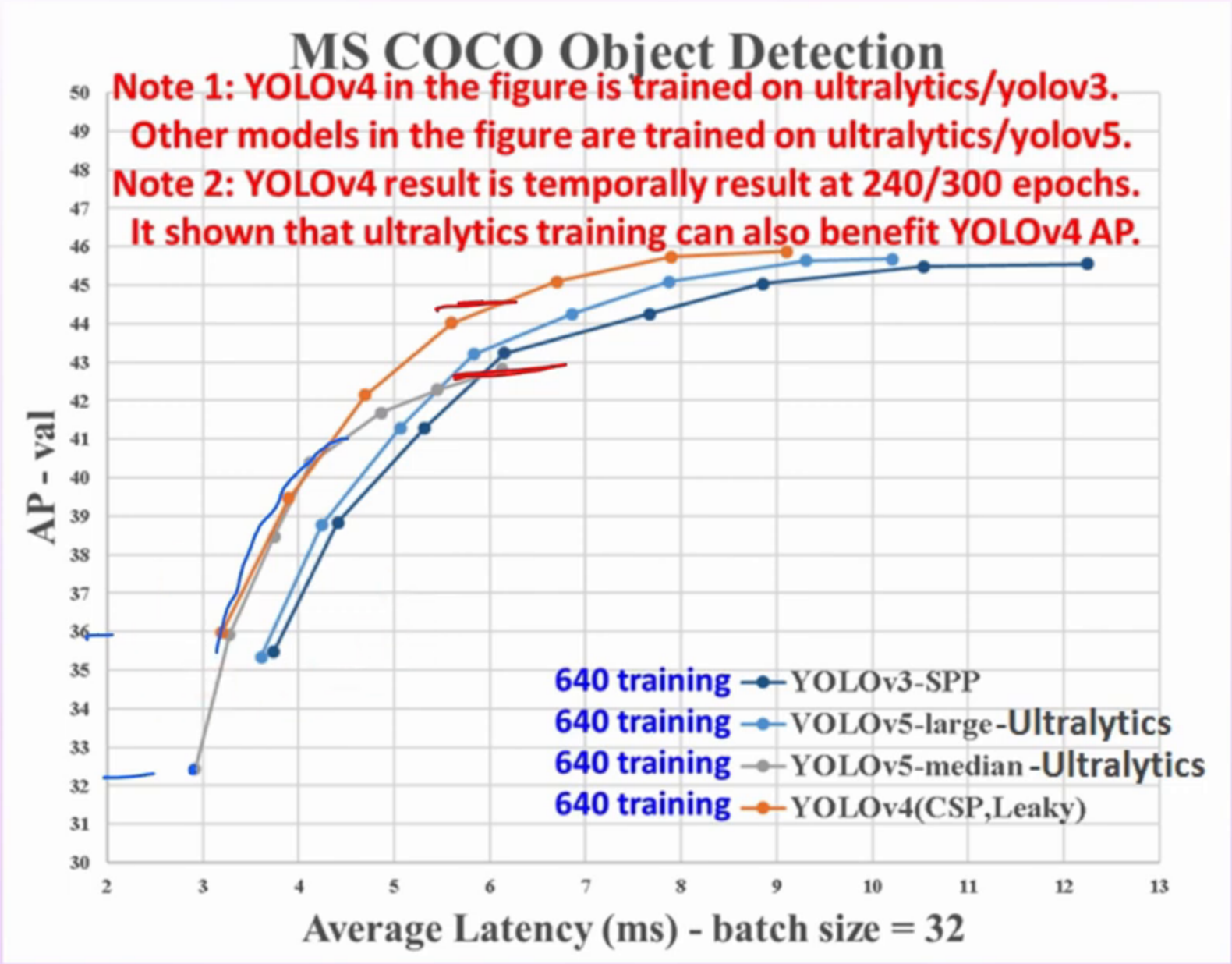
- The latency shouldn't be measured with batch=32. The latency must be measured with batch=1, because the higher batch higher latency. The latency is the time of a complete data processing cycle, it cannot be less than processing a whole batch can take up to 1 second depends on batch-size

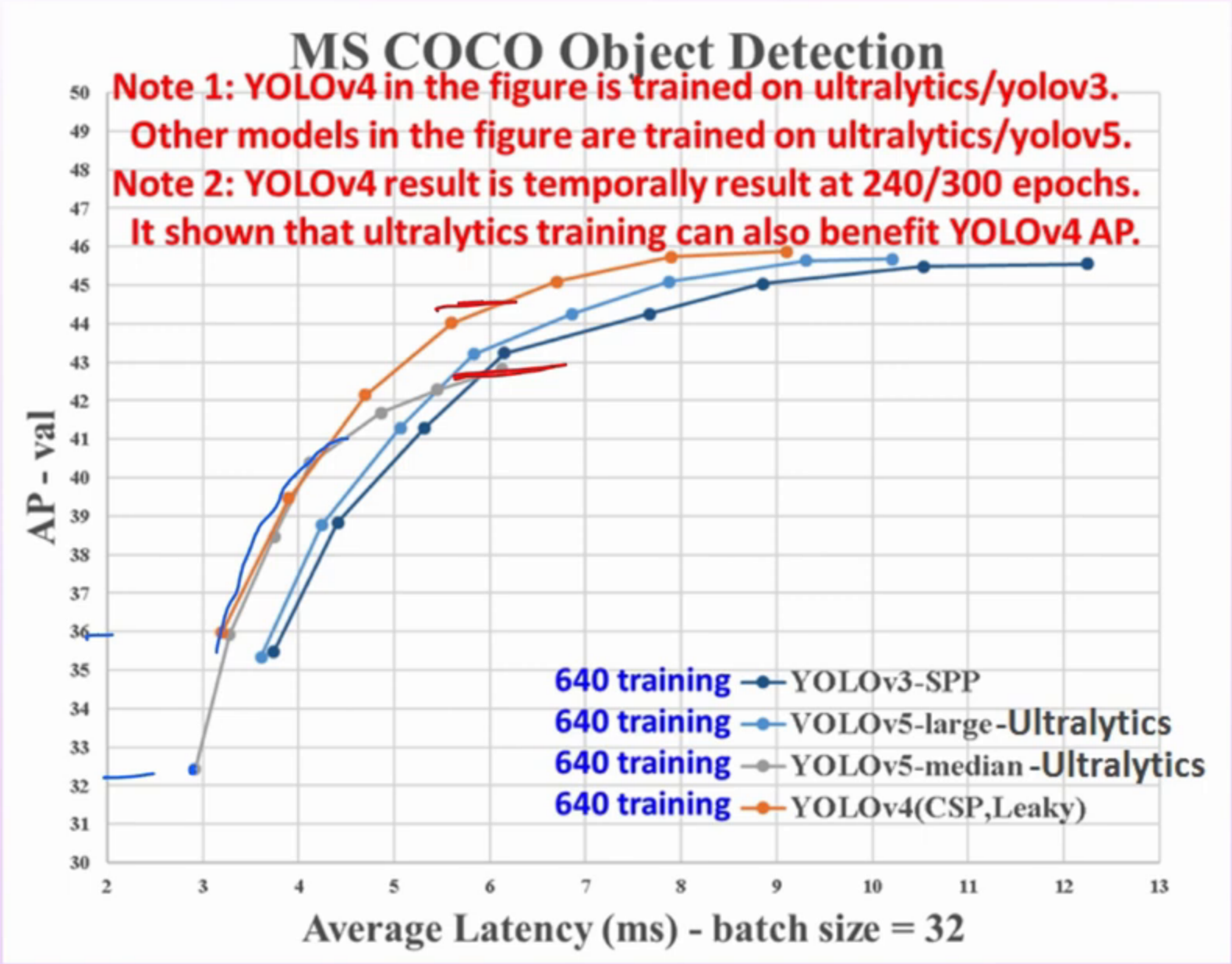
- If there is used batch=32 for both Yolov5 vs EfficientDet (I don't know), then this is ok, but only for Yolov5 vs EfficientDet, ar FPS (not for latency), it can't be compared with any other results where is batch=1

- Size of weights: yolov5x.pt - `366 MB` , yolov5s.pt - `27 MB`


MS COCO Object Detection
Note 1: YOLOv4 in the figure is trained on ultralytics/yolov3.
Other models in the figure are trained on ultralytics/yolov5.
Note 2: YOLOv4 result is temporally result at 240/300 epochs.
It shown that ultralytics training can also benefit YOLOv4 AP.
AP - val
Average Latency (ms) - batch size = 32
640 training YOLOv3-SPP
640 training YOLOv5-large -Ultralytics
640 training YOLOv5-median -Ultralytics
640 training YOLOv4(CSP,Leaky)


MS COCO Object Detection
Note 1: YOLOv4 in the figure is trained on ultralytics/yolov3.
Other models in the figure are trained on ultralytics/yolov5.
Note 2: YOLOv4 result is temporally result at 240/300 epochs.
It shown that ultralytics training can also benefit YOLOv4 AP.
AP - val
Average Latency (ms) - batch size = 32
640 training YOLOv3-SPP
640 training YOLOv5-large -Ultralytics
640 training YOLOv5-median -Ultralytics
640 training YOLOv4(CSP,Leaky)


MS COCO Object Detection
Note 1: YOLOv4 in the figure is trained on ultralytics/yolov3.
Other models in the figure are trained on ultralytics/yolov5.
Note 2: YOLOv4 result is temporally result at 240/300 epochs.
It shown that ultralytics training can also benefit YOLOv4 AP.
AP - val
Average Latency (ms) - batch size = 32
640 training YOLOv3-SPP
640 training YOLOv5-large-Ultralytics
640 training YOLOv5-median-Ultralytics
640 training YOLOv4(CSP,Leaky)


MS COCO Object Detection
Note 1: YOLOv4 in the figure is trained on ultralytics/yolov3.
Other models in the figure are trained on ultralytics/yolov5.
Note 2: YOLOv4 result is temporally result at 240/300 epochs.
It shown that ultralytics training can also benefit YOLOv4 AP.
AP - val
Average Latency (ms) - batch size = 32
640 training YOLOv3-SPP
640 training YOLOv5-large-Ultralytics
640 training YOLOv5-median-Ultralytics
640 training YOLOv4(CSP,Leaky)

Ultralytics Response

glenn-jocher on Jun 11, 2020 · edited by glenn-jocher

Edits ▾ Member ···

@amusi and all others regarding publication and naming:

Thank you for your feedback! Regarding publication, we would very much like to be able to publish a paper detailing the various modifications employed to achieve these results (among all 3 major components: architecture, loss function and training methodology), however we are extremely limited on resources, and thus must smartly select how best to deploy these in order to keep our business viable as a going concern, while also continuing our work of pushing the boundaries of what's possible in this and other fields.

Importantly **these models are neither static nor complete** at this time. Our recent open-sourcing of this work is simply part of our ongoing research, and is not any sort of final product, and for this reason it is not accompanied by any publication. Our current goal is to continue internal R&D throughout the remainder of 2020, and hopefully open source and publish at least a short synopsis of this to Arxiv by the end of the year.

Regarding naming, we do take note of the comments surrounding the issue. YOLOv5 is an internal designation assigned to this work, which is now open-sourced. The exact name employed here is not a concern for us (we are open to alternatives!), we are instead focused on producing, improving, and delivering results to our clients, and to the open-source community by extension when our contract terms allow for it, which we push for often.

We appreciate all feedback, and we will try to be as responsive as we can going forward!

☺ 👍 31 👎 14 ❤️ 15

Ultralytics Response

glenn-jocher on Jun 11, 2020 · edited by glenn-jocher

Edits ▾ Member ···

@amusi and all others regarding publication and naming:

Thank you for your feedback! Regarding publication, we would very much like to be able to publish a paper detailing the various modifications employed to achieve these results (among all 3 major components: architecture, loss function and training methodology), however we are extremely limited on resources, and thus must smartly select how best to deploy these in order to keep our business viable as a going concern, while also continuing our work of pushing the boundaries of what's possible in this and other fields.

Importantly **these models are neither static nor complete** at this time. Our recent open-sourcing of this work is simply part of our ongoing research, and is not any sort of final product, and for this reason it is not accompanied by any publication. Our current goal is to continue internal R&D throughout the remainder of 2020, and hopefully open source and publish at least a short synopsis of this to Arxiv by the end of the year.

Regarding naming, we do take note of the comments surrounding the issue. YOLOv5 is an internal designation assigned to this work, which is now open-sourced. The exact name employed here is not a concern for us (we are open to alternatives!), we are instead focused on producing, improving, and delivering results to our clients, and to the open-source community by extension when our contract terms allow for it, which we push for often.

We appreciate all feedback, and we will try to be as responsive as we can going forward!

Ultralytics Response

glenn-jocher on Jun 11, 2020 · edited by glenn-jocher

Edits ▾ Member ···

@amusi and all others regarding publication and naming:

Thank you for your feedback! Regarding publication, we would very much like to be able to publish a paper detailing the various modifications employed to achieve these results (among all 3 major components: architecture, loss function and training methodology), however we are extremely limited on resources, and thus must smartly select how best to deploy these in order to keep our business viable as a going concern, while also continuing our work of pushing the boundaries of what's possible in this and other fields.

Importantly **these models are neither static nor complete** at this time. Our recent open-sourcing of this work is simply part of our ongoing research, and is not any sort of final product, and for this reason it is not accompanied by any publication. Our current goal is to continue internal R&D throughout the remainder of 2020, and hopefully open source and publish at least a short synopsis of this to Arxiv by the end of the year.

Regarding naming, we do take note of the comments surrounding the issue. YOLOv5 is an internal designation assigned to this work, which is now open-sourced. The exact name employed here is not a concern for us (we are open to alternatives!), we are instead focused on producing, improving, and delivering results to our clients, and to the open-source community by extension when our contract terms allow for it, which we push for often.

We appreciate all feedback, and we will try to be as responsive as we can going forward!

☺ 👍 31 👎 14 ❤️ 15

Ultralytics Response

glenn-jocher on Jun 11, 2020 · edited by glenn-jocher

Edits ▾ Member ···

@amusi and all others regarding publication and naming:

Thank you for your feedback! Regarding publication, we would very much like to be able to publish a paper detailing the various modifications employed to achieve these results (among all 3 major components: architecture, loss function and training methodology), however we are extremely limited on resources, and thus must smartly select how best to deploy these in order to keep our business viable as a going concern, while also continuing our work of pushing the boundaries of what's possible in this and other fields.

Importantly **these models are neither static nor complete** at this time. Our recent open-sourcing of this work is simply part of our ongoing research, and is not any sort of final product, and for this reason it is not accompanied by any publication. Our current goal is to continue internal R&D throughout the remainder of 2020, and hopefully open source and publish at least a short synopsis of this to Arxiv by the end of the year.

Regarding naming, we do take note of the comments surrounding the issue. YOLOv5 is an internal designation assigned to this work, which is now open-sourced. The exact name employed here is not a concern for us (we are open to alternatives!), we are instead focused on producing, improving, and delivering results to our clients, and to the open-source community by extension when our contract terms allow for it, which we push for often.

We appreciate all feedback, and we will try to be as responsive as we can going forward!

☺ 👍 31 👎 14 ❤️ 15

Ultralytics Response

glenn-jocher on Jun 11, 2020 · edited by glenn-jocher

Edits ▾ Member ···

@amusi and all others regarding publication and naming:

Thank you for your feedback! Regarding publication, we would very much like to be able to publish a paper detailing the various modifications employed to achieve these results (among all 3 major components: architecture, loss function and training methodology), however we are extremely limited on resources, and thus must smartly select how best to deploy these in order to keep our business viable as a going concern, while also continuing our work of pushing the boundaries of what's possible in this and other fields.

Importantly **these models are neither static nor complete** at this time. Our recent open-sourcing of this work is simply part of our ongoing research, and is not any sort of final product, and for this reason it is not accompanied by any publication. Our current goal is to continue internal R&D throughout the remainder of 2020, and hopefully open source and publish at least a short synopsis of this to Arxiv by the end of the year.

Regarding naming, we do take note of the comments surrounding the issue. YOLOv5 is an internal designation assigned to this work, which is now open-sourced. The exact name employed here is not a concern for us (we are open to alternatives!), we are instead focused on producing, improving, and delivering results to our clients, and to the open-source community by extension when our contract terms allow for it, which we push for often.

We appreciate all feedback, and we will try to be as responsive as we can going forward!

☺ 👍 31 👎 14 ❤️ 15

Ultralytics Response

glenn-jocher on Jun 11, 2020 · edited by glenn-jocher

Edits ▾ Member ···

@amusi and all others regarding publication and naming:

Thank you for your feedback! Regarding publication, we would very much like to be able to publish a paper detailing the various modifications employed to achieve these results (among all 3 major components: architecture, loss function and training methodology), however we are extremely limited on resources, and thus must smartly select how best to deploy these in order to keep our business viable as a going concern, while also continuing our work of pushing the boundaries of what's possible in this and other fields.

Importantly **these models are neither static nor complete** at this time. Our recent open-sourcing of this work is simply part of our ongoing research, and is not any sort of final product, and for this reason it is not accompanied by any publication. Our current goal is to continue internal R&D throughout the remainder of 2020, and hopefully open source and publish at least a short synopsis of this to Arxiv by the end of the year.

Regarding naming, we do take note of the comments surrounding the issue. YOLOv5 is an internal designation assigned to this work, which is now open-sourced. The exact name employed here is not a concern for us (we are open to alternatives!), we are instead focused on producing, improving, and delivering results to our clients, and to the open-source community by extension when our contract terms allow for it, which we push for often.

We appreciate all feedback, and we will try to be as responsive as we can going forward!

☺ 👍 31 👎 14 ❤️ 15

Ultralytics Response

glenn-jocher on Jun 11, 2020 · edited by glenn-jocher

Edits ▾ Member ···

@amusi and all others regarding publication and naming:

Thank you for your feedback! Regarding publication, we would very much like to be able to publish a paper detailing the various modifications employed to achieve these results (among all 3 major components: architecture, loss function and training methodology), however we are extremely limited on resources, and thus must smartly select how best to deploy these in order to keep our business viable as a going concern, while also continuing our work of pushing the boundaries of what's possible in this and other fields.

Importantly **these models are neither static nor complete** at this time. Our recent open-sourcing of this work is simply part of our ongoing research, and is not any sort of final product, and for this reason it is not accompanied by any publication. Our current goal is to continue internal R&D throughout the remainder of 2020, and hopefully open source and publish at least a short synopsis of this to Arxiv by the end of the year.

Regarding naming, we do take note of the comments surrounding the issue. YOLOv5 is an internal designation assigned to this work, which is now open-sourced. The exact name employed here is not a concern for us (we are open to alternatives!), we are instead focused on producing, improving, and delivering results to our clients, and to the open-source community by extension when our contract terms allow for it, which we push for often.

We appreciate all feedback, and we will try to be as responsive as we can going forward!

☺ 👍 31 👎 14 ❤️ 15

Ultralytics Response

glenn-jocher on Jun 11, 2020 · edited by glenn-jocher

Edits ▾ Member ···

@amusi and all others regarding publication and naming:

Thank you for your feedback! Regarding publication, we would very much like to be able to publish a paper detailing the various modifications employed to achieve these results (among all 3 major components: architecture, loss function and training methodology), however we are extremely limited on resources, and thus must smartly select how best to deploy these in order to keep our business viable as a going concern, while also continuing our work of pushing the boundaries of what's possible in this and other fields.

Importantly **these models are neither static nor complete** at this time. Our recent open-sourcing of this work is simply part of our ongoing research, and is not any sort of final product, and for this reason it is not accompanied by any publication. Our current goal is to continue internal R&D throughout the remainder of 2020, and hopefully open source and publish at least a short synopsis of this to Arxiv by the end of the year.

Regarding naming, we do take note of the comments surrounding the issue. YOLOv5 is an internal designation assigned to this work, which is now open-sourced. The exact name employed here is not a concern for us (we are open to alternatives!), we are instead focused on producing, improving, and delivering results to our clients, and to the open-source community by extension when our contract terms allow for it, which we push for often.

We appreciate all feedback, and we will try to be as responsive as we can going forward!

☺ 👍 31 👎 14 ❤️ 15

Ultralytics Response

glenn-jocher on Jun 11, 2020 · edited by glenn-jocher

Edits ▾ Member ···

@amusi and all others regarding publication and naming:

Thank you for your feedback! Regarding publication, we would very much like to be able to publish a paper detailing the various modifications employed to achieve these results (among all 3 major components: architecture, loss function and training methodology), however we are extremely limited on resources, and thus must smartly select how best to deploy these in order to keep our business viable as a going concern, while also continuing our work of pushing the boundaries of what's possible in this and other fields.

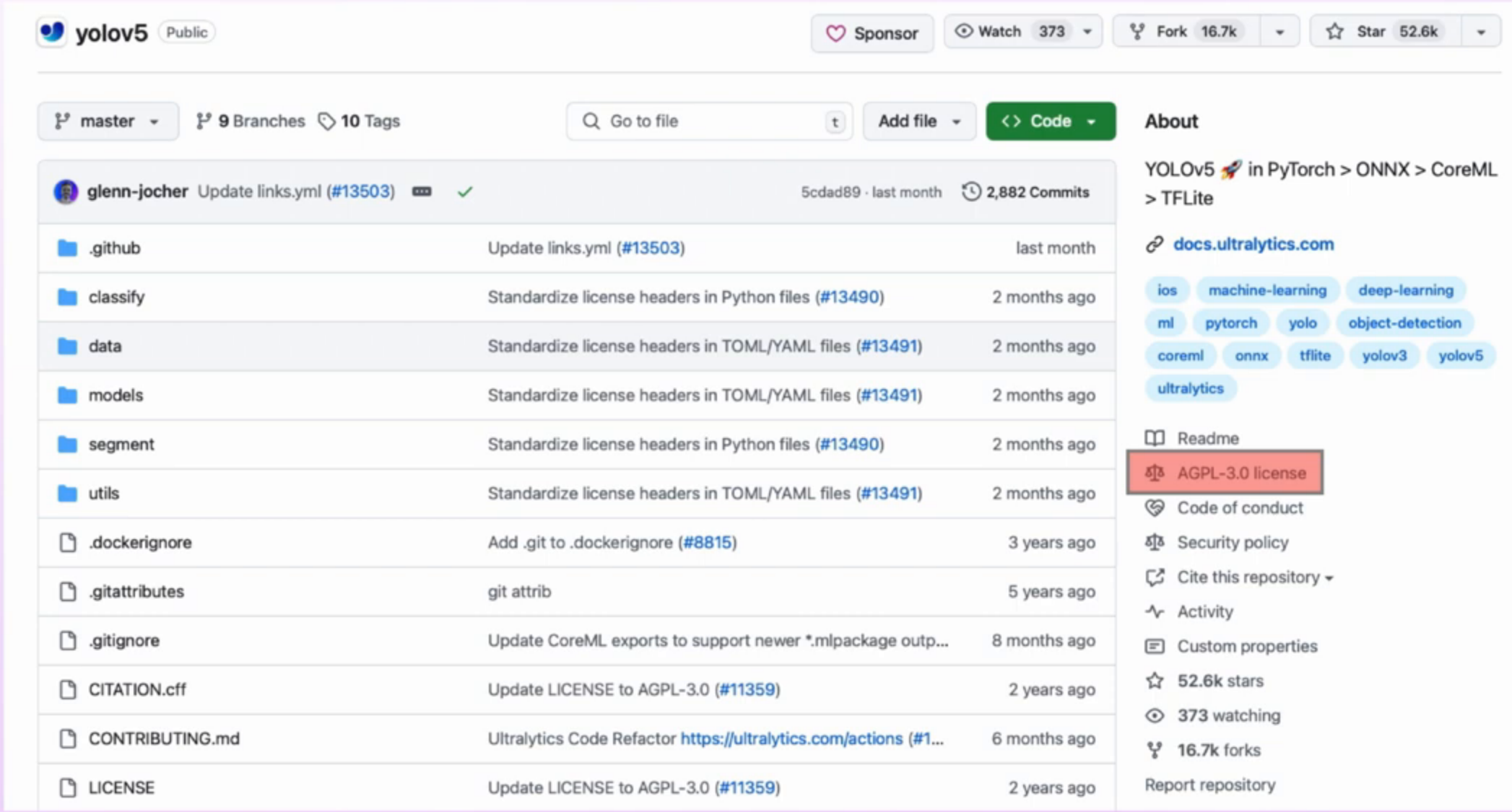
Importantly **these models are neither static nor complete** at this time. Our recent open-sourcing of this work is simply part of our ongoing research, and is not any sort of final product, and for this reason it is not accompanied by any publication. Our current goal is to continue internal R&D throughout the remainder of 2020, and hopefully open source and publish at least a short synopsis of this to Arxiv by the end of the year.

Regarding naming, we do take note of the comments surrounding the issue. YOLOv5 is an internal designation assigned to this work, which is now open-sourced. The exact name employed here is not a concern for us (we are open to alternatives!), we are instead focused on producing, improving, and delivering results to our clients, and to the open-source community by extension when our contract terms allow for it, which we push for often.

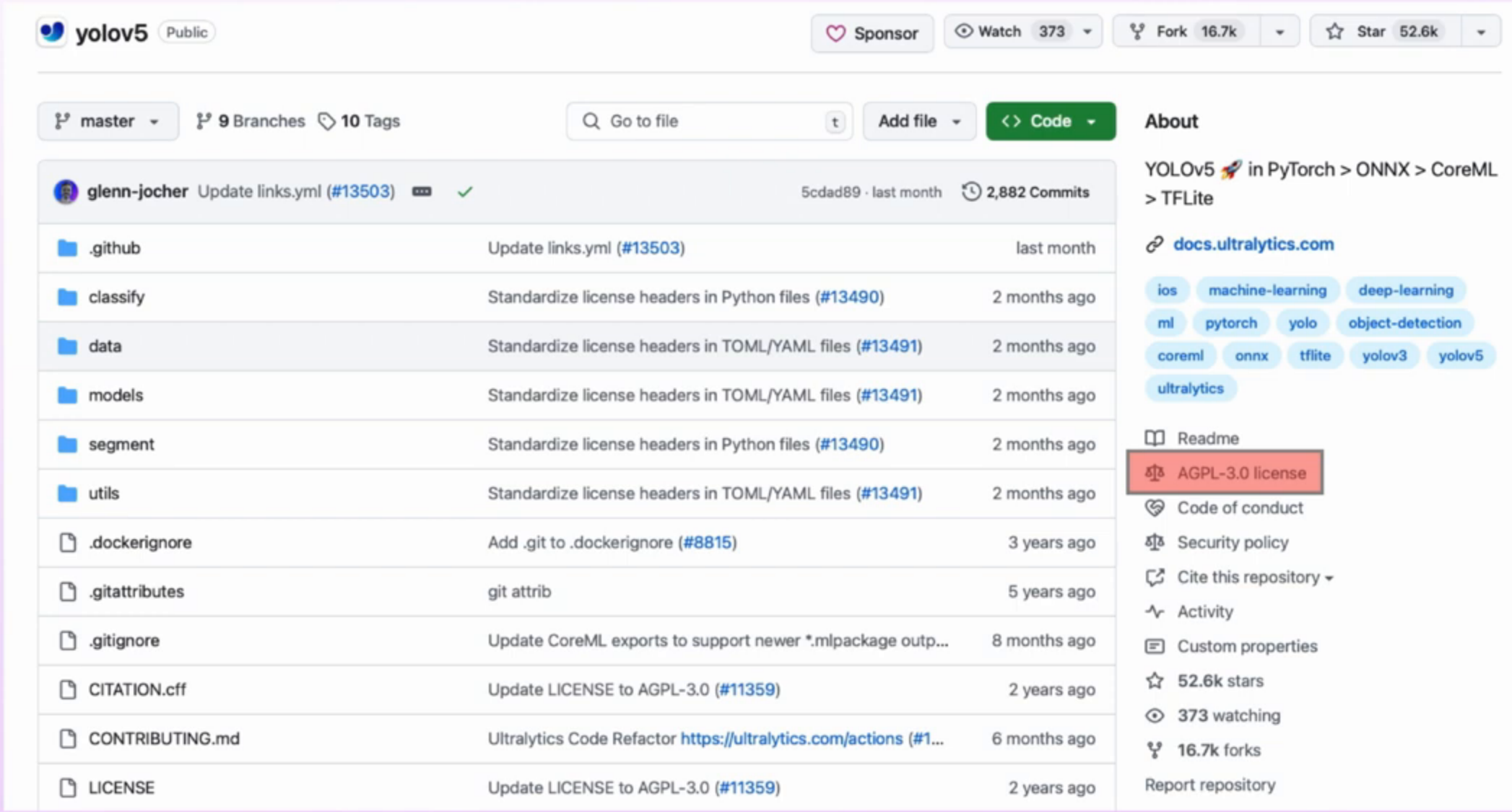
We appreciate all feedback, and we will try to be as responsive as we can going forward!

☺ 👍 31 👎 14 ❤️ 15

Ultralytics License

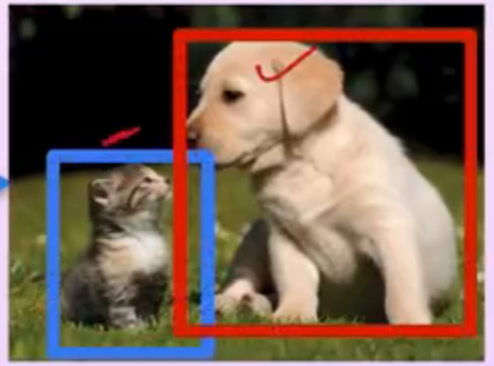


Ultralytics License




YOLO-V5

YOLO-V5



YOLO-V5

Backbone

CSP
Darknet53

Neck

SPP
+
PAN

Head

YOLOv3
Head

YOLO-V5

Backbone

CSP Darknet53

Neck

SPP

+

PAN

Head

YOLOv3 Head

YOLO-V5

Backbone

CSP
Darknet53

Neck

SPP
+
PAN

Head

YOLOv3
Head

YOLO-V5

Backbone

3 x 3

6 x 6

Mish

Swish

YOLO-V5

Backbone

3 x 3 → 6 x 6

Mish → Swish

Add/Remove CSP blocks

YOLO-V5

Backbone

Modified
CSP
Darknet53

Neck

SPP
+
PAN

YOLO-V5

Backbone

Modified
CSP
Darknet53

Neck

SPP → SPPF

PAN → CSP-PAN

YOLO-V5

Backbone

Modified
CSP
Darknet53

Neck

SPP → SPPF

PAN → CSP-PAN

YOLO-V5

Backbone

Modified CSP Darknet53

Neck

SPPF + CSP-PAN

Head

Modified YOLOv3 Head

YOLO-V5

Backbone

Modified
CSP
Darknet53

Neck

SPPF
+
CSP-PAN

Head

Modified
YOLOv3
Head

YOLO-V5

Backbone

Modified CSP Darknet53

Neck

SPPF + CSP-PAN

Head

Modified YOLOv3 Head

Darknet

Languages

- C 62.0%
- Cuda 14.6%
- C++ 12.5%
- Python 4.5%
- PowerShell 2.9%
- CMake 1.7%
- Other 1.8%

YoloV5

About

YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite

Languages

- Python 98.5%
- Shell 1.1%
- Dockerfile 0.4%

Darknet

Languages

- C 62.0%
- Cuda 14.6%
- C++ 12.5%
- Python 4.5%
- PowerShell 2.9%
- CMake 1.7%
- Other 1.8%

YoloV5

About

YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite

Languages

- Python 98.5%
- Shell 1.1%
- Dockerfile 0.4%

Darknet

Languages

- C 62.0%
- Cuda 14.6%
- C++ 12.5%
- Python 4.5%
- PowerShell 2.9%
- CMake 1.7%
- Other 1.8%

YoloV5

About

YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite

Languages

- Python 98.5%
- Shell 1.1%
- Dockerfile 0.4%

Darknet

Languages

- C 62.0%
- Cuda 14.6%
- C++ 12.5%
- Python 4.5%
- PowerShell 2.9%
- CMake 1.7%
- Other 1.8%

YoloV5

About

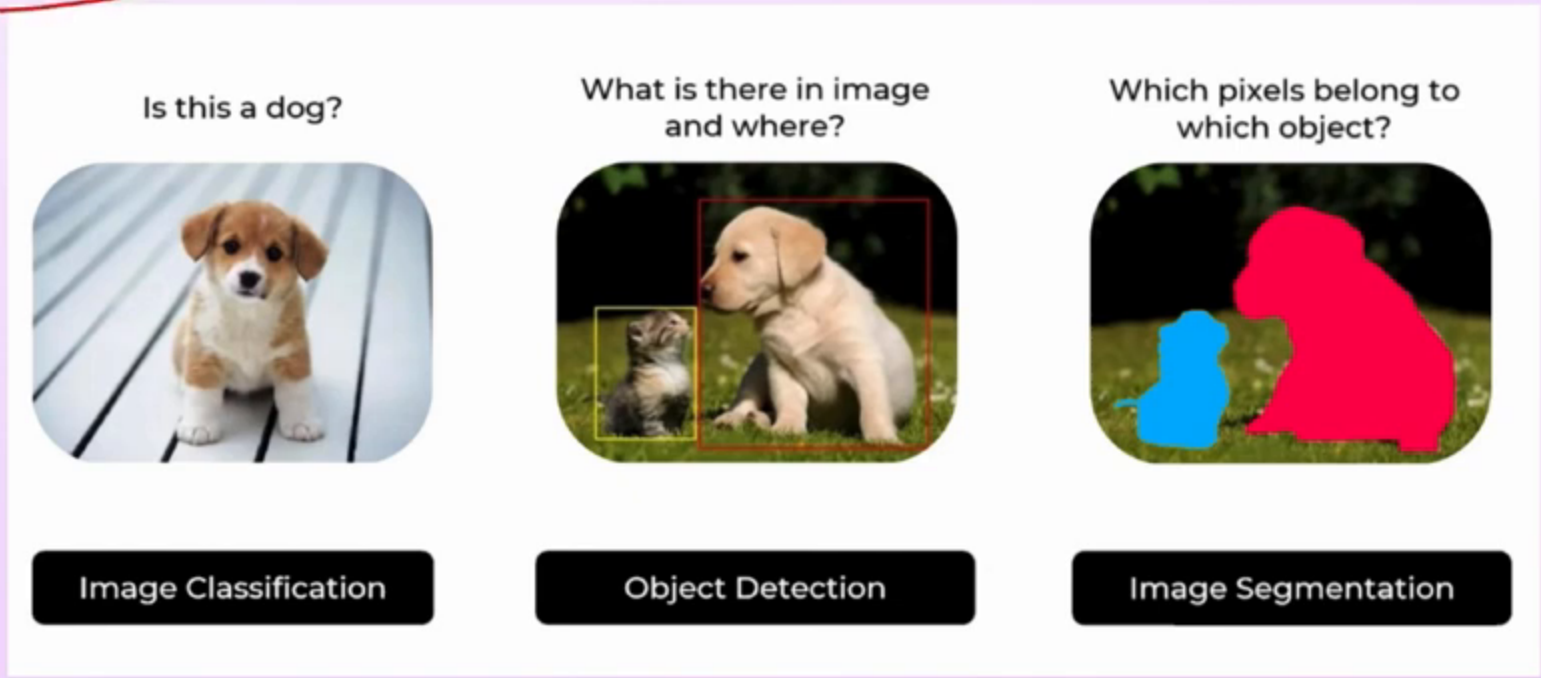
YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite

Languages

- Python 98.5%
- Shell 1.1%
- Dockerfile 0.4%

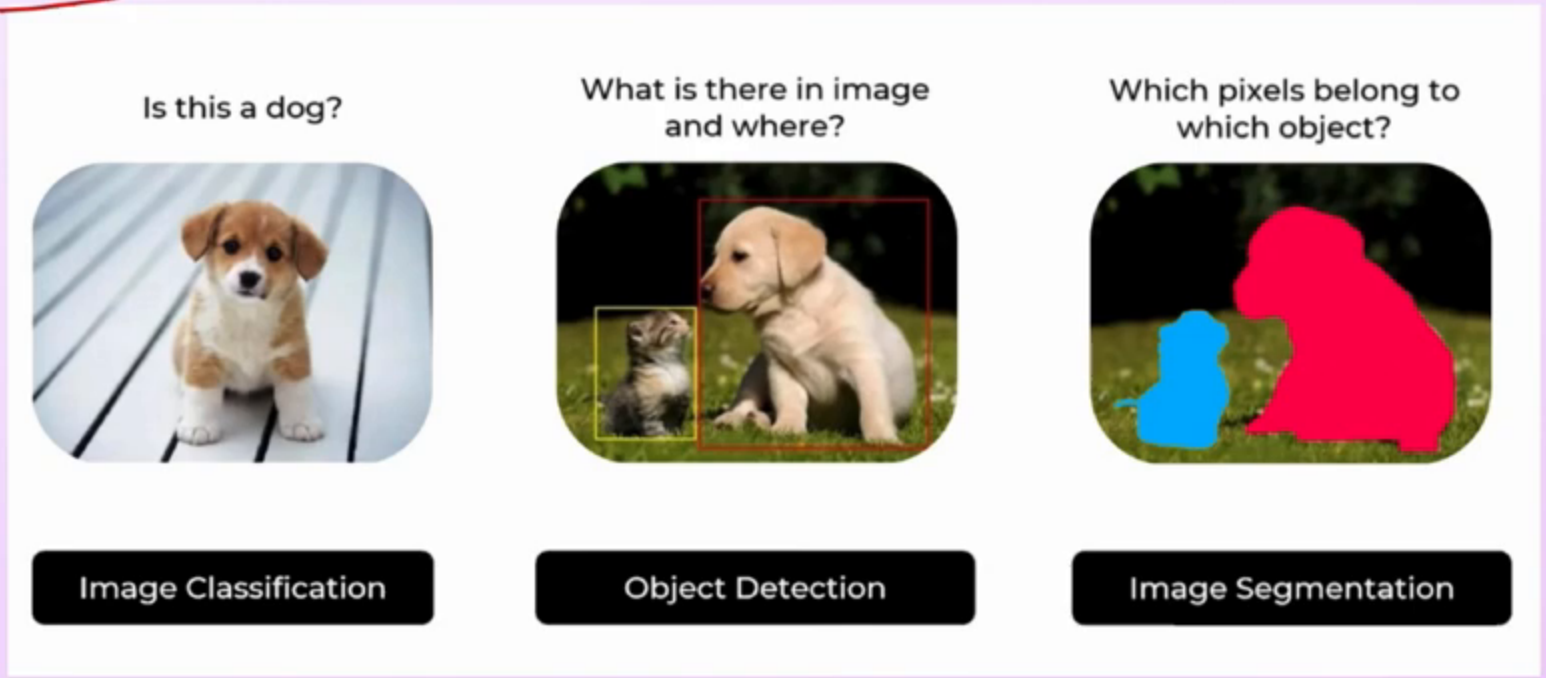
Features of Yolo-v5 Repo

* **Models**
  * Object Detection
  * Classification
  * InstanceSegmentation



Features of Yolo-v5 Repo

* **Models**
  * Object Detection
  * Classification
  * InstanceSegmentation



Features of Yolo-v5 Repo

* **Models**
  * Object Detection
  * Classification
  * InstanceSegmentation

* **Training & FineTuning**
  * Multi-Scale Training
  * Mixed Precision Training
  * GA for Hyperparameters
  * Experiment Tracking
  * AutoAnchor
  * Lr schedulers
  * EMA (Exponential Moving Average)

Features of Yolo-v5 Repo

* **Models**
  * Object Detection
  * Classification
  * InstanceSegmentation

* **Training & FineTuning**
  * Multi-Scale Training
  * Mixed Precision Training
  * GA for Hyperparameters
  * Experiment Tracking
  * AutoAnchor
  * Lr schedulers
  * EMA (Exponential Moving Average)

Features of Yolo-v5 Repo

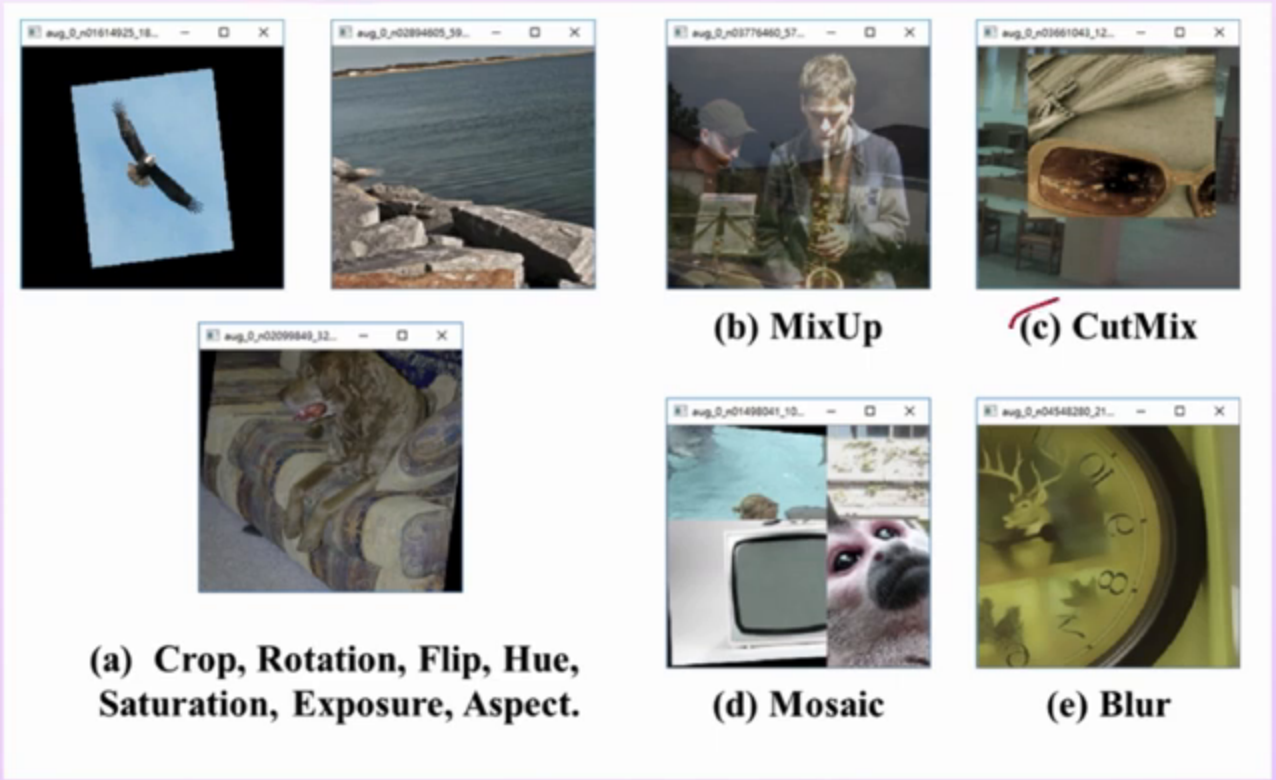
* **Models**
  * Object Detection
  * Classification
  * InstanceSegmentation

* **Training & FineTuning**
  * Multi-Scale Training
  * Mixed Precision Training
  * GA for Hyperparameters
  * Experiment Tracking
  * AutoAnchor
  * Lr schedulers
  * EMA (Exponential Moving Average)

Features of Yolo-v5 Repo

* **Models**
  * Object Detection
  * Classification
  * InstanceSegmentation
* **Training & FineTuning**
  * Multi-Scale Training
  * Mixed Precision Training
  * GA for Hyperparameters
  * Experiment Tracking
  * AutoAnchor
  * Lr schedulers
  * EMA (Exponential Moving Average)

* **Data Augmentations**
  * Mosaic
  * Mixup
  * Albumentations



Features of Yolo-v5 Repo

* **Models**
  * Object Detection
  * Classification
  * InstanceSegmentation
* **Training & FineTuning**
  * Multi-Scale Training
  * Mixed Precision Training
  * GA for Hyperparameters
  * Experiment Tracking
  * AutoAnchor
  * Lr schedulers
  * EMA (Exponential Moving Average)

* **Data Augmentations**
  * Mosaic
  * Mixup
  * Albumentations

Features of Yolo-v5 Repo

* **Models**
  * Object Detection
  * Classification
  * InstanceSegmentation
* **Training & FineTuning**
  * Multi-Scale Training
  * Mixed Precision Training
  * GA for Hyperparameters
  * Experiment Tracking
  * AutoAnchor
  * Lr schedulers
  * EMA (Exponential Moving Average)
* **Data Augmentations**
  * Mosaic
  * Mixup
  * Albumentations

* **Deployment**
  * ONNX
  * TensorRT
  * OpenVINO
  * CoreML
  * TF.js & TFLite
* **Ease of use**
  * One click prediction
    * Webcams, RTSP, Videos, Images
  * Easy Training & Deployment
* **Community**
  * Well documented & regularly updated
  * Frequent updates for Pretrained weights
  * Large community support

Features of Yolo-v5 Repo

* **Models**
  * Object Detection
  * Classification
  * InstanceSegmentation
* **Training & FineTuning**
  * Multi-Scale Training
  * Mixed Precision Training
  * GA for Hyperparameters
  * Experiment Tracking
  * AutoAnchor
  * Lr schedulers
  * EMA (Exponential Moving Average)
* **Data Augmentations**
  * Mosaic
  * Mixup
  * Albumentations

* **Deployment**
  * ONNX
  * TensorRT
  * OpenVINO
  * CoreML
  * TF.js & TFLite
* **Ease of use**
  * One click prediction
    * Webcams, RTSP, Videos, Images
  * Easy Training & Deployment
* **Community**
  * Well documented & regularly updated
  * Frequent updates for Pretrained weights
  * Large community support

Features of Yolo-v5 Repo

* **Models**
  * Object Detection
  * Classification
  * InstanceSegmentation
* **Training & FineTuning**
  * Multi-Scale Training
  * Mixed Precision Training
  * GA for Hyperparameters
  * Experiment Tracking
  * AutoAnchor
  * Lr schedulers
  * EMA (Exponential Moving Average)
* **Data Augmentations**
  * Mosaic
  * Mixup
  * Albumentations

* **Deployment**
  * ONNX
  * TensorRT
  * OpenVINO
  * CoreML
  * TF.js & TFLite
* **Ease of use**
  * One click prediction
    * Webcams, RTSP, Videos, Images
  * Easy Training & Deployment
* **Community**
  * Well documented & regularly updated
  * Frequent updates for Pretrained weights
  * Large community support

Features of Yolo-v5 Repo

* **Models**
  * Object Detection
  * Classification
  * InstanceSegmentation
* **Training & FineTuning**
  * Multi-Scale Training
  * Mixed Precision Training
  * GA for Hyperparameters
  * Experiment Tracking
  * AutoAnchor
  * Lr schedulers
  * EMA (Exponential Moving Average)
* **Data Augmentations**
  * Mosaic
  * Mixup
  * Albumentations

* **Deployment**
  * ONNX
  * TensorRT
  * OpenVINO
  * CoreML
  * TF.js & TFLite
* **Ease of use**
  * One click prediction
    * Webcams, RTSP, Videos, Images
  * Easy Training & Deployment
* **Community**
  * Well documented & regularly updated
  * Frequent updates for Pretrained weights
  * Large community support

Features of Yolo-v5 Repo

* **Models**
  * Object Detection
  * Classification
  * InstanceSegmentation
* **Training & FineTuning**
  * Multi-Scale Training
  * Mixed Precision Training
  * GA for Hyperparameters
  * Experiment Tracking
  * AutoAnchor
  * Lr schedulers
  * EMA (Exponential Moving Average)
* **Data Augmentations**
  * Mosaic
  * Mixup
  * Albumentations

* **Deployment**
  * ONNX
  * TensorRT
  * OpenVINO
  * CoreML
  * TF.js & TFLite
* **Ease of use**
  * One click prediction
    * Webcams, RTSP, Videos, Images
  * Easy Training & Deployment
* **Community**
  * Well documented & regularly updated
  * Frequent updates for Pretrained weights
  * Large community support

Features of Yolo-v5 Repo

* **Models**
  * Object Detection
  * Classification
  * InstanceSegmentation
* **Training & FineTuning**
  * Multi-Scale Training
  * Mixed Precision Training
  * GA for Hyperparameters
  * Experiment Tracking
  * AutoAnchor
  * Lr schedulers
  * EMA (Exponential Moving Average)
* **Data Augmentations**
  * Mosaic
  * Mixup
  * Albumentations

* **Deployment**
  * ONNX
  * TensorRT
  * OpenVINO
  * CoreML
  * TF.js & TFLite
* **Ease of use**
  * One click prediction
    * Webcams, RTSP, Videos, Images
  * Easy Training & Deployment
* **Community**
  * Well documented & regularly updated
  * Frequent updates for Pretrained weights
  * Large community support

Features of Yolo-v5 Repo

- **Models**
  - Object Detection
  - Classification
  - InstanceSegmentation
- **Training & FineTuning**
  - Multi-Scale Training
  - Mixed Precision Training
  - GA for Hyperparameters
  - Experiment Tracking
  - AutoAnchor
  - Lr schedulers
  - EMA (Exponential Moving Average)
- **Data Augmentations**
  - Mosaic
  - Mixup
  - Albumentations

- **Deployment**
  - ONNX
  - TensorRT
  - OpenVINO
  - CoreML
  - TF.js & TFLite
- **Ease of use**
  - One click prediction
    - Webcams, RTSP, Videos, Images
  - Easy Training & Deployment
- **Community**
  - Well documented & regularly updated
  - Frequent updates for Pretrained weights
  - Large community support

Features of Yolo-v5 Repo

- **Models**
  - Object Detection
  - Classification
  - InstanceSegmentation
- **Training & FineTuning**
  - Multi-Scale Training
  - Mixed Precision Training
  - GA for Hyperparameters
  - Experiment Tracking
  - AutoAnchor
  - Lr schedulers
  - EMA (Exponential Moving Average)
- **Data Augmentations**
  - Mosaic
  - Mixup
  - Albumentations

- **Deployment**
  - ONNX
  - TensorRT
  - OpenVINO
  - CoreML
  - TF.js & TFLite
- **Ease of use**
  - One click prediction
    - Webcams, RTSP, Videos, Images
  - Easy Training & Deployment
- **Community**
  - Well documented & regularly updated
  - Frequent updates for Pretrained weights
  - Large community support

YOLO-V5

Backbone

Modified
CSP
Darknet53

Neck

SPPF
+
CSP-PAN

Head

Modified
YOLOv3
Head

YOLO-V5

Backbone

Modified
CSP
Darknet53

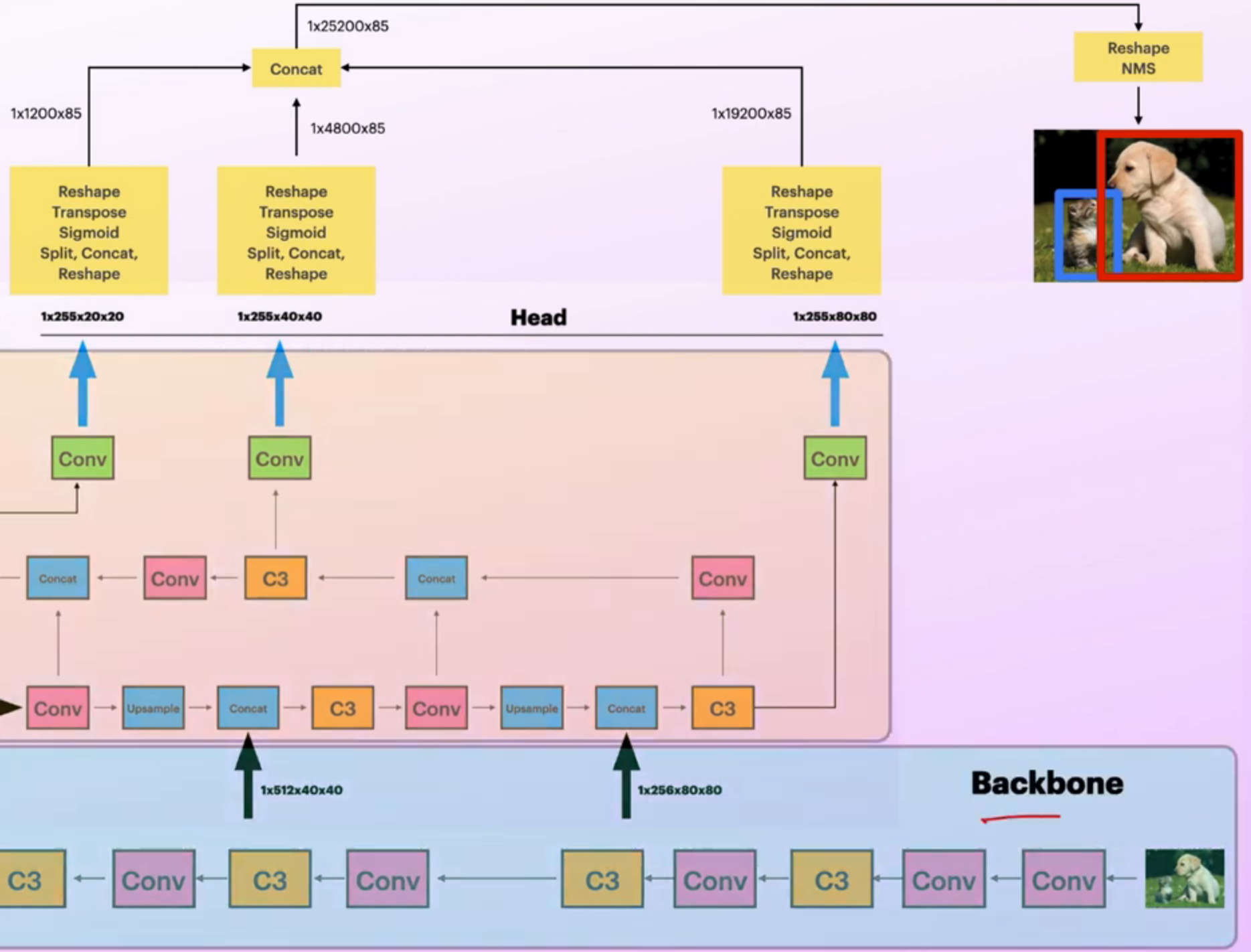
Neck

SPPF
+
CSP-PAN

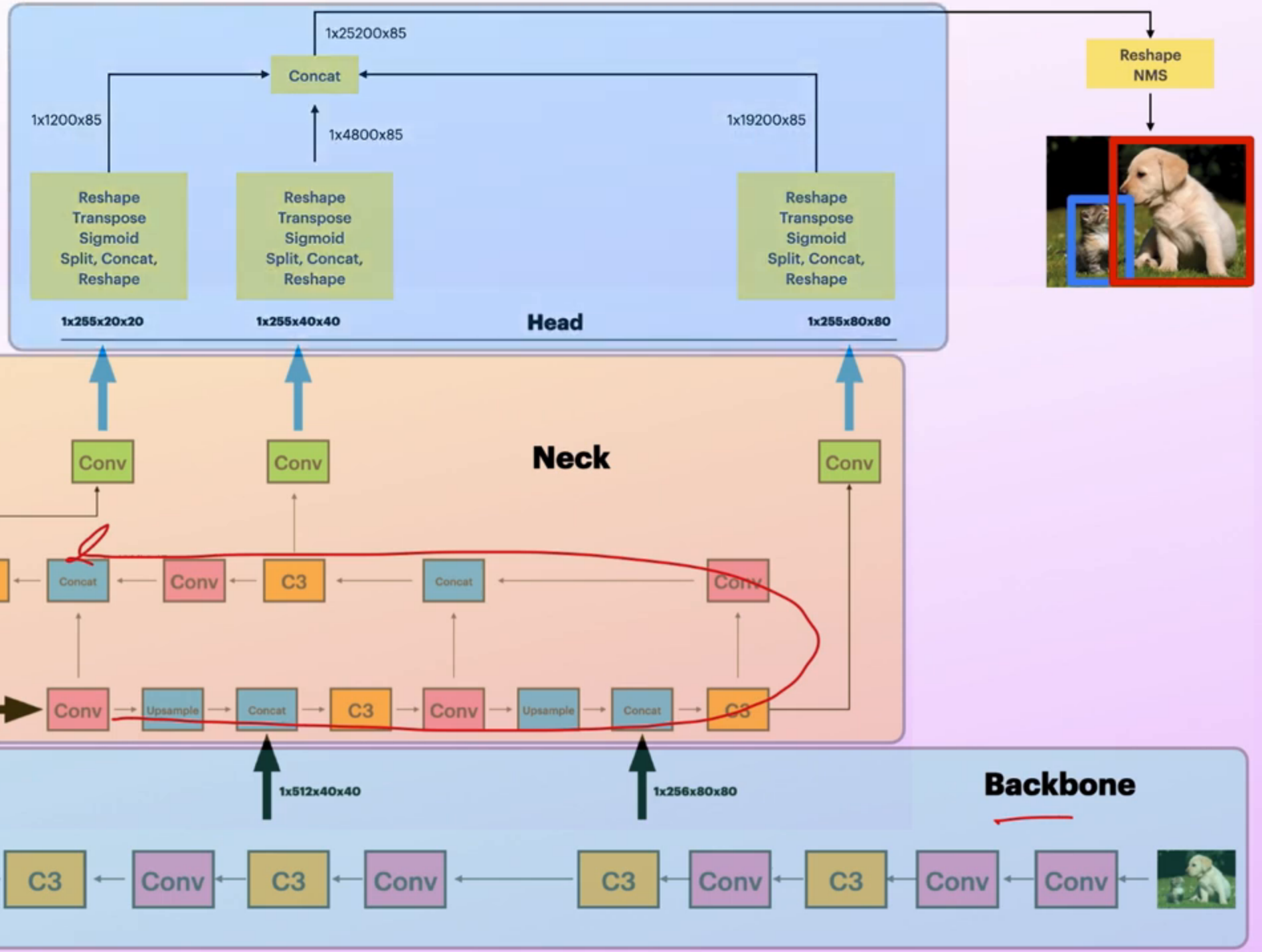
Head

Modified
YOLOv3
Head

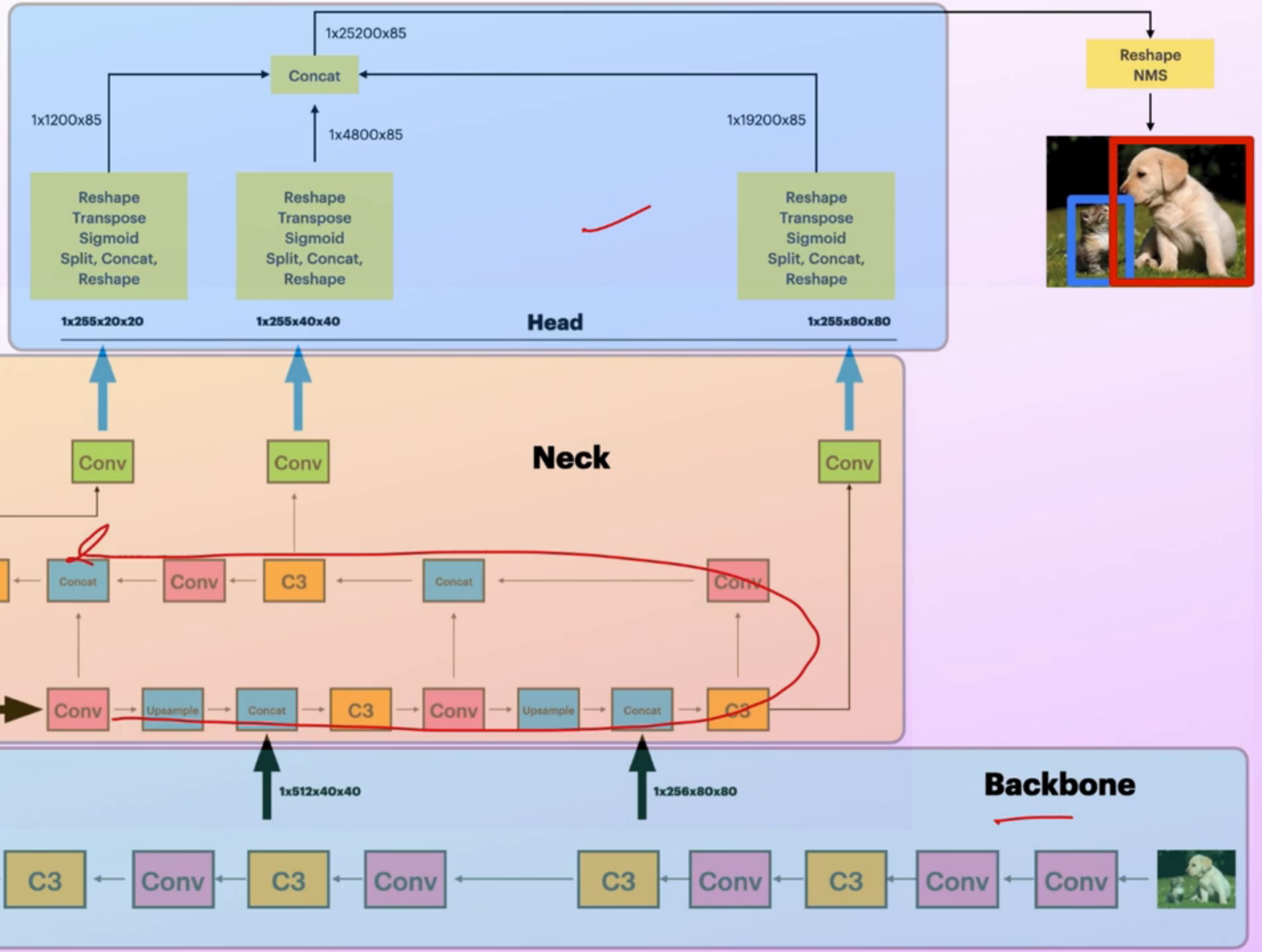
YOLO V5


1x25200x85
Concat
Reshape
NMS
1x1200x85
1x4800x85
1x19200x85
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
1x255x20x20
1x255x40x40
Head
1x255x80x80
Conv
Conv
Conv
C3
Concat
Conv
C3
Concat
Conv
Conv
Upsample
Concat
C3
Conv
Upsample
Concat
C3
1x1024x20x20
1x512x40x40
1x256x80x80
Backbone
SPPF
C3
Conv
C3
Conv
C3
Conv
C3
Conv
Conv

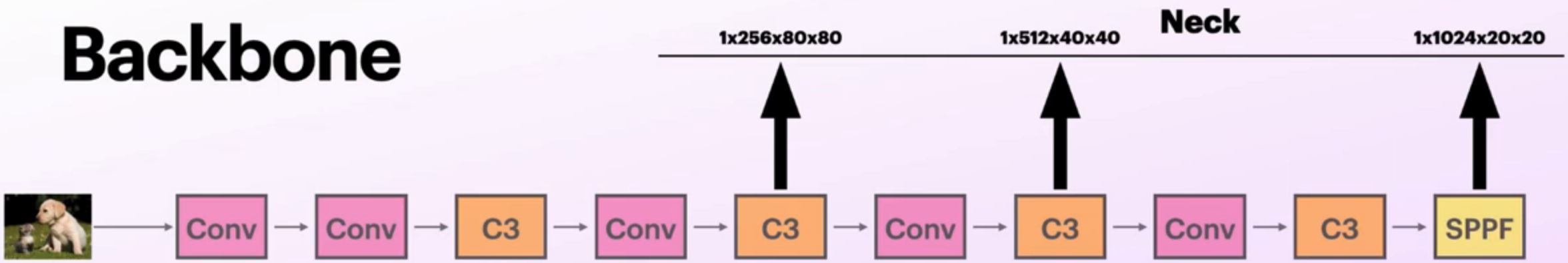
YOLO V5


1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
1x255x20x20
1x255x40x40
Head
1x255x80x80
Reshape
NMS
Conv
Conv
Neck
Conv
C3
Concat
Conv
C3
Concat
Conv
Conv
Upsample
Concat
C3
Conv
Upsample
Concat
C3
1x1024x20x20
1x512x40x40
1x256x80x80
Backbone
SPPF
C3
Conv
C3
Conv
C3
Conv
C3
Conv
Conv

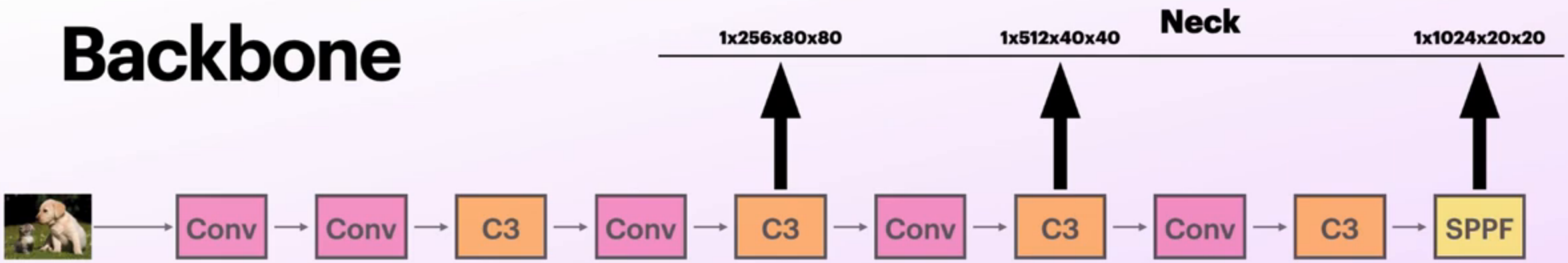
YOLO V5


1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
1x255x20x20
1x255x40x40
Head
1x255x80x80
Reshape
NMS
Conv
Conv
Neck
Conv
C3
Concat
Conv
C3
Concat
Conv
Conv
Upsample
Concat
C3
Conv
Upsample
Concat
C3
1x1024x20x20
1x512x40x40
1x256x80x80
Backbone
SPPF
C3
Conv
C3
Conv
C3
Conv
C3
Conv
Conv

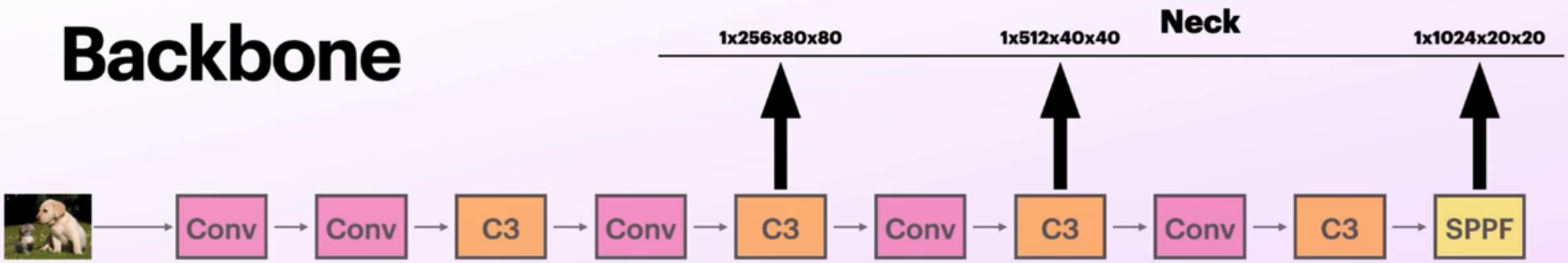
Backbone


Neck
1x256x80x80
1x512x40x40
1x1024x20x20
Conv
Conv
C3
Conv
C3
Conv
C3
Conv
C3
SPPF

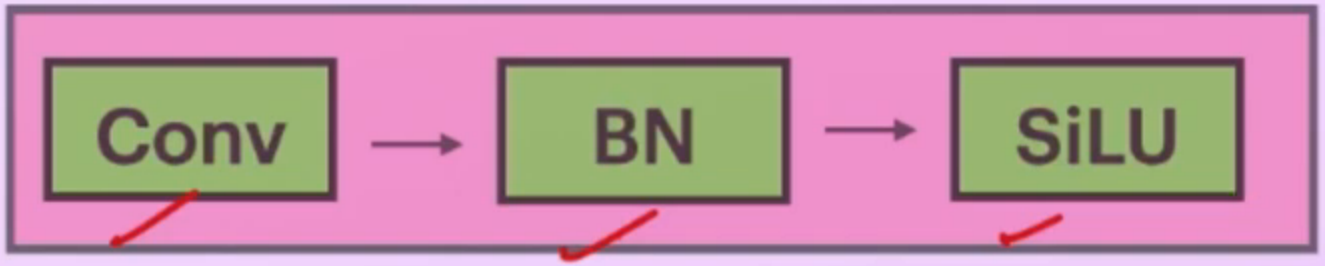
Backbone


Neck
1x256x80x80
1x512x40x40
1x1024x20x20
Conv
Conv
C3
Conv
C3
Conv
C3
Conv
C3
SPPF

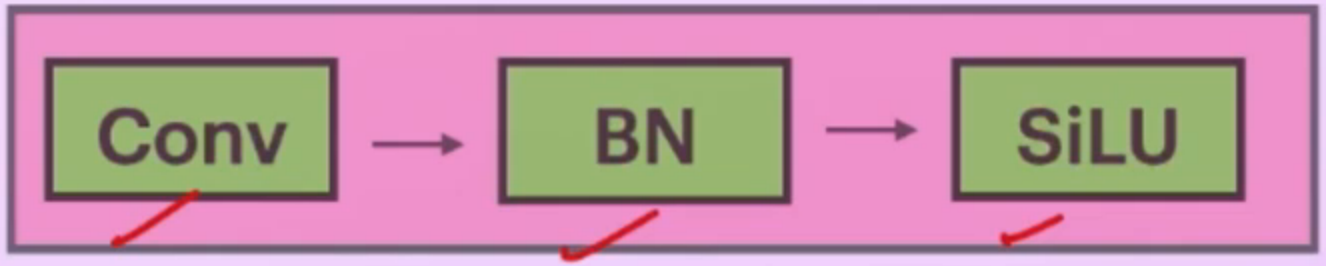
Backbone


Neck
1x256x80x80
1x512x40x40
1x1024x20x20
Conv
Conv
C3
Conv
C3
Conv
C3
Conv
C3
SPPF


Conv
BN
SiLU



$$SiLU = x * sigmoid(x)$$



$$SiLU = x * sigmoid(x)$$

Conv → Conv → C3 → Conv → C3 → Conv → C

Conv → Conv → C3 → Conv → C3 → Conv

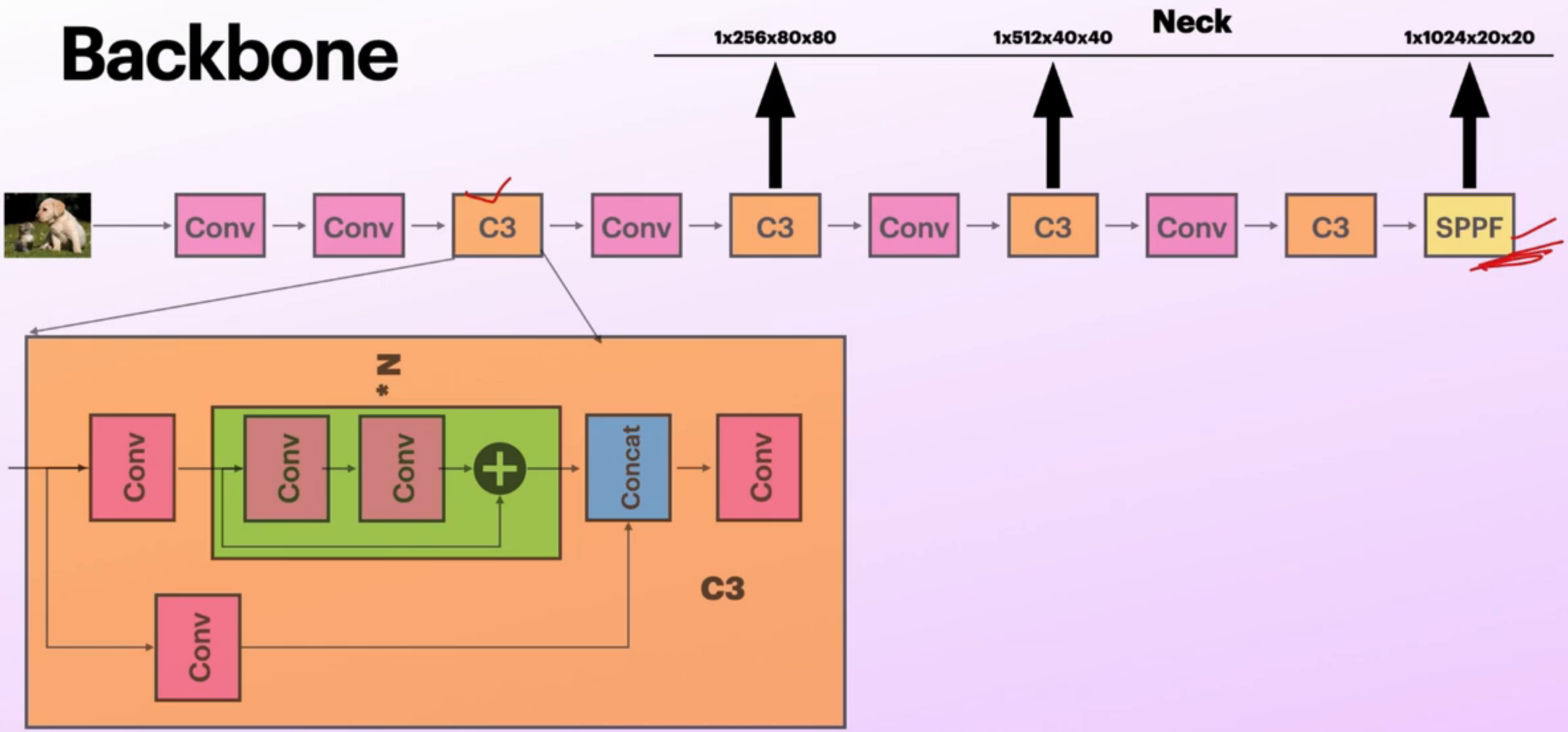
n *

Conv → Conv → Conv → (+) → Concat → Conv

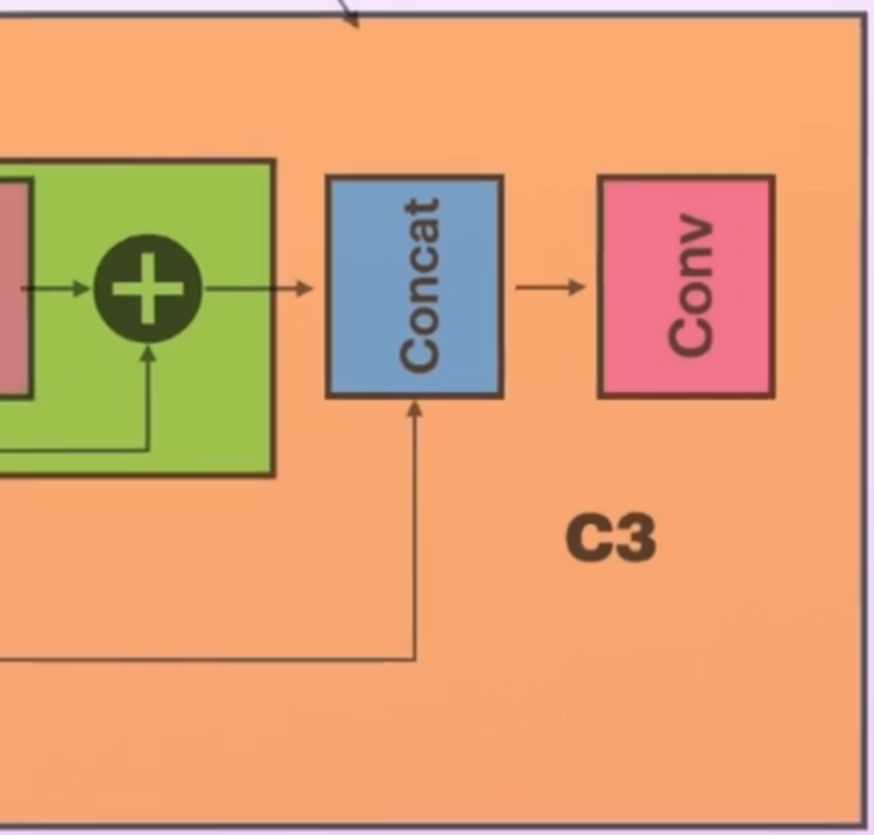
Conv

C3

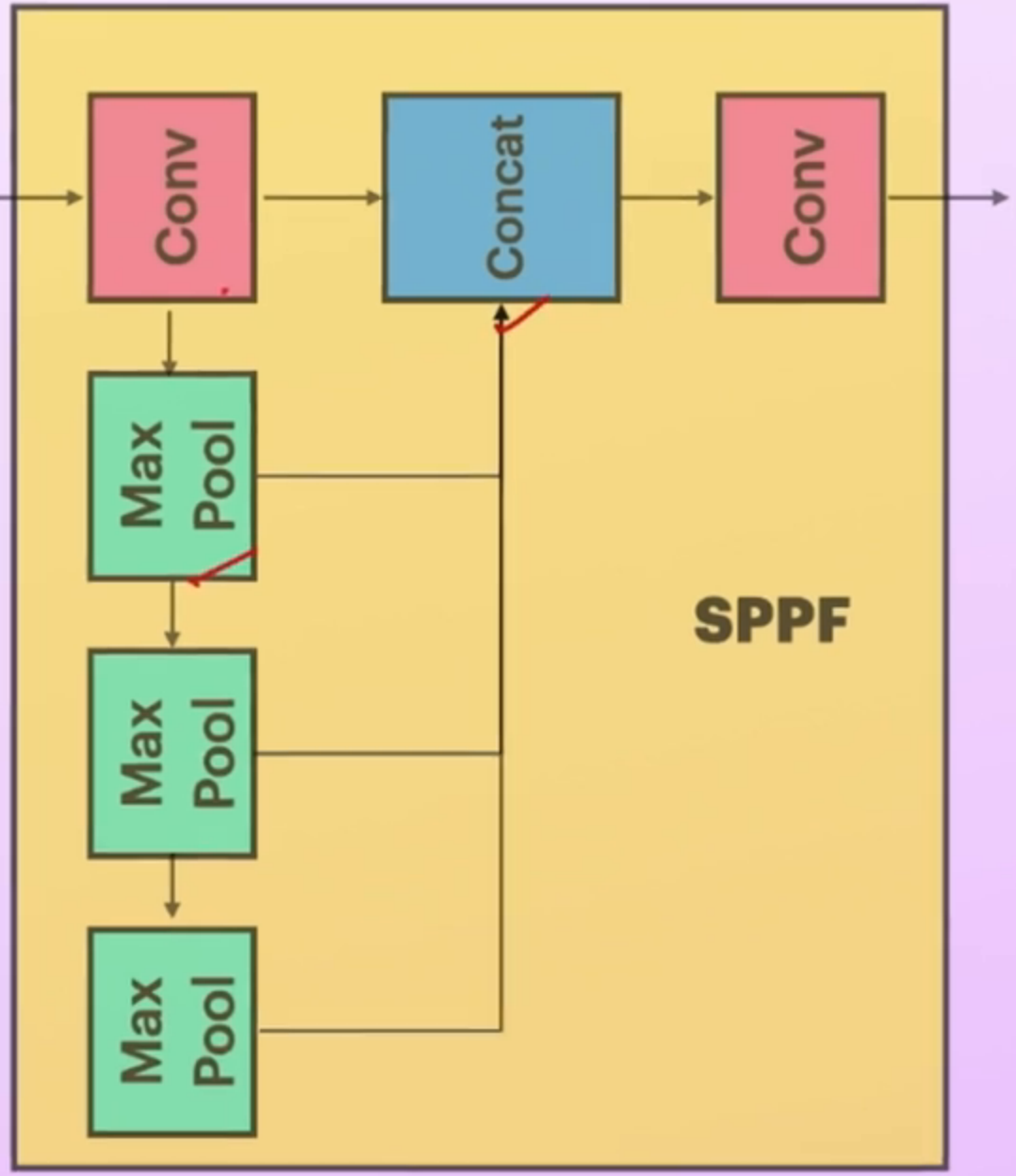
Backbone



$$SiLU = x * sigmoid(x)$$



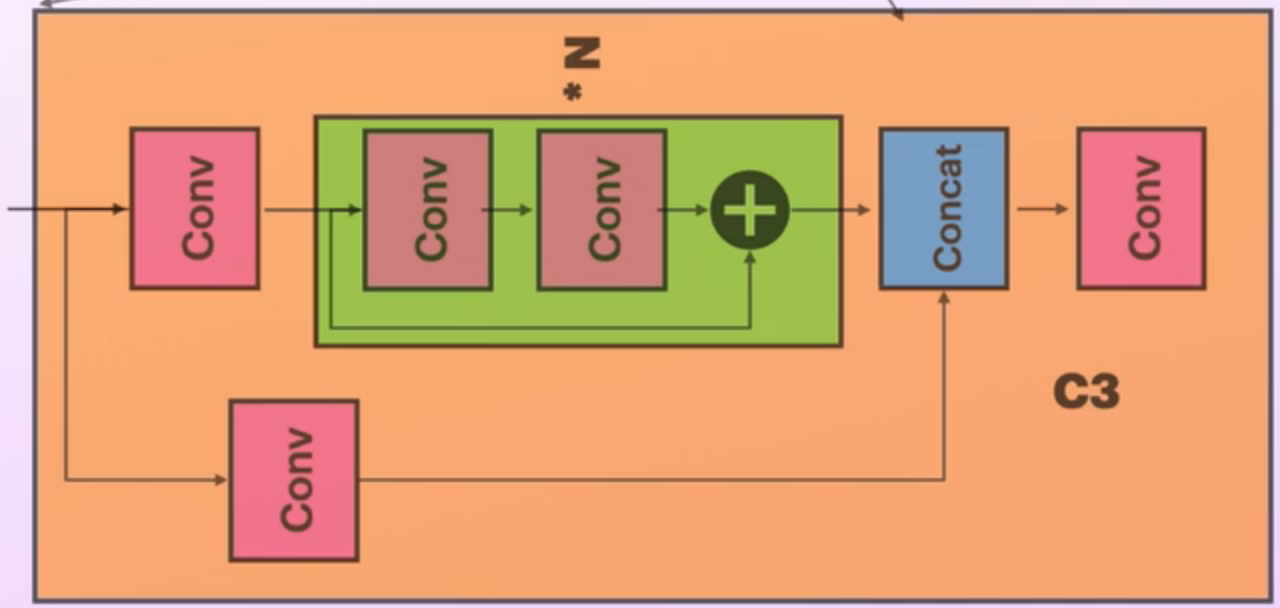
$$SiLU = x * sigmoid(x)$$

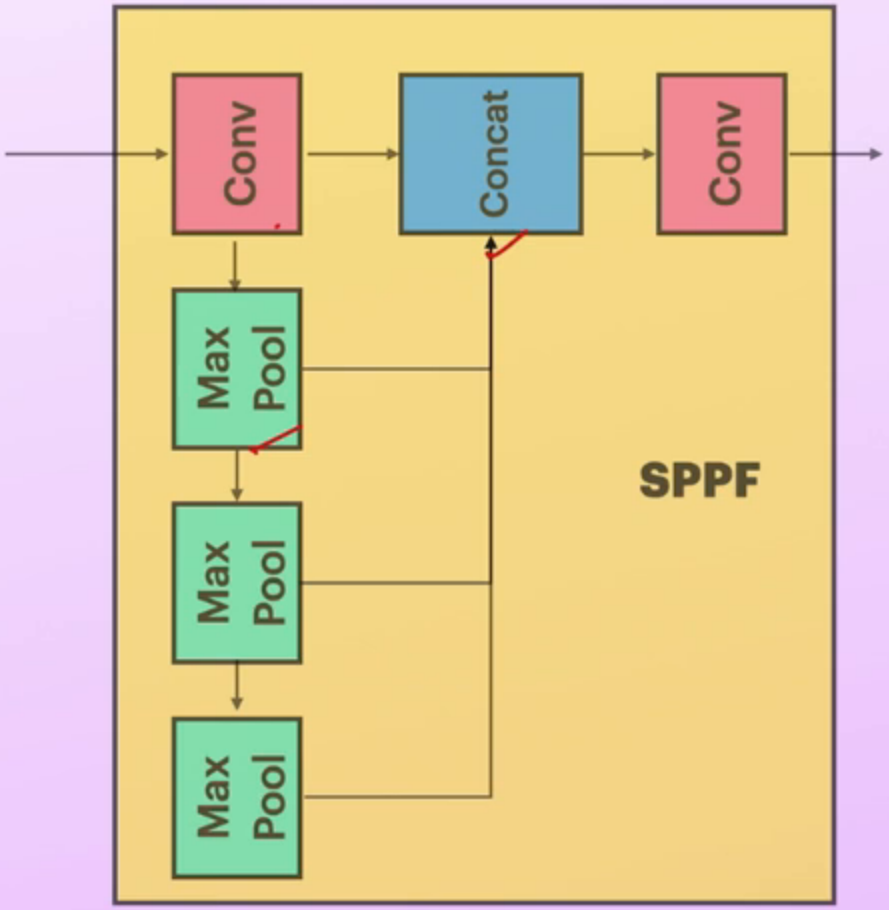


Backbone

Conv → Conv → C3 → Conv → C3 → Conv → C3 → Conv → C3 → SPPF

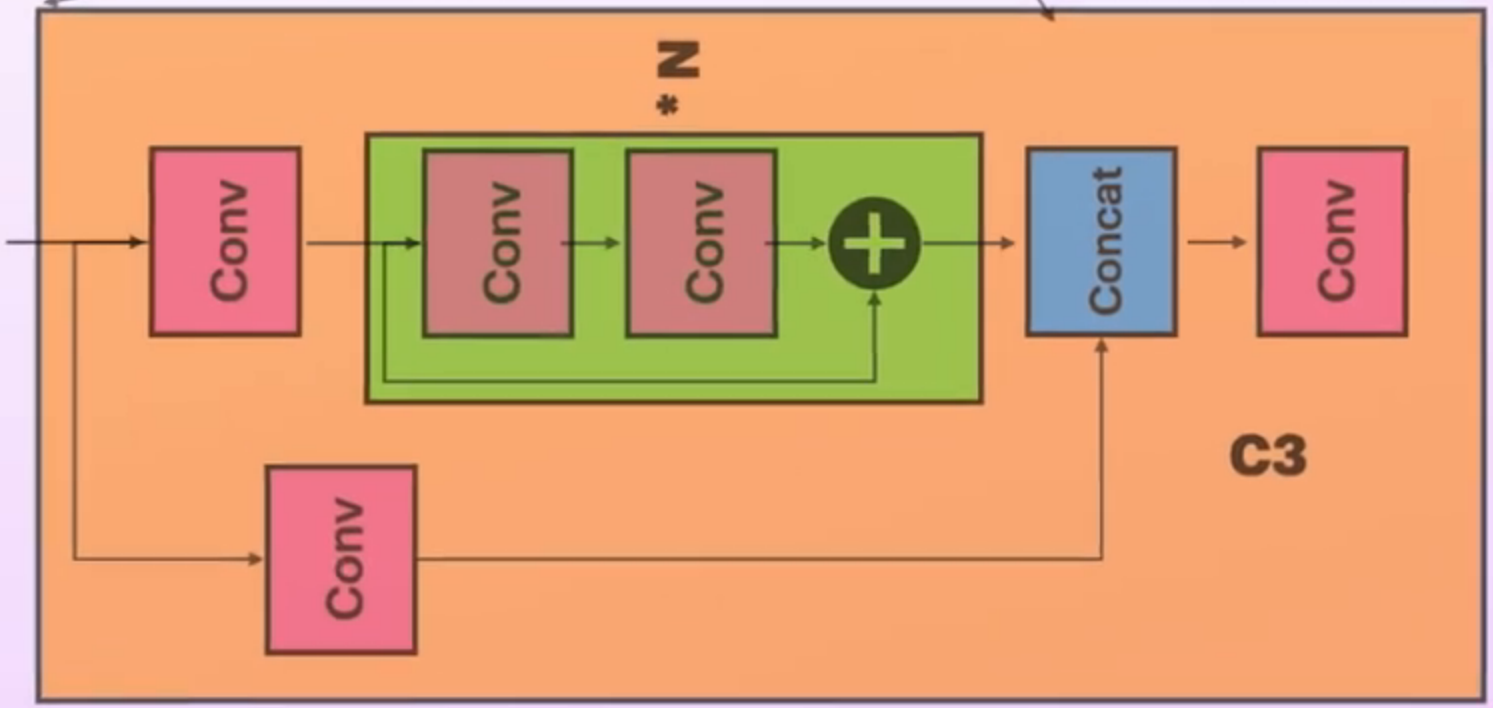
Neck: 1x256x80x80, 1x512x40x40, 1x1024x20x20

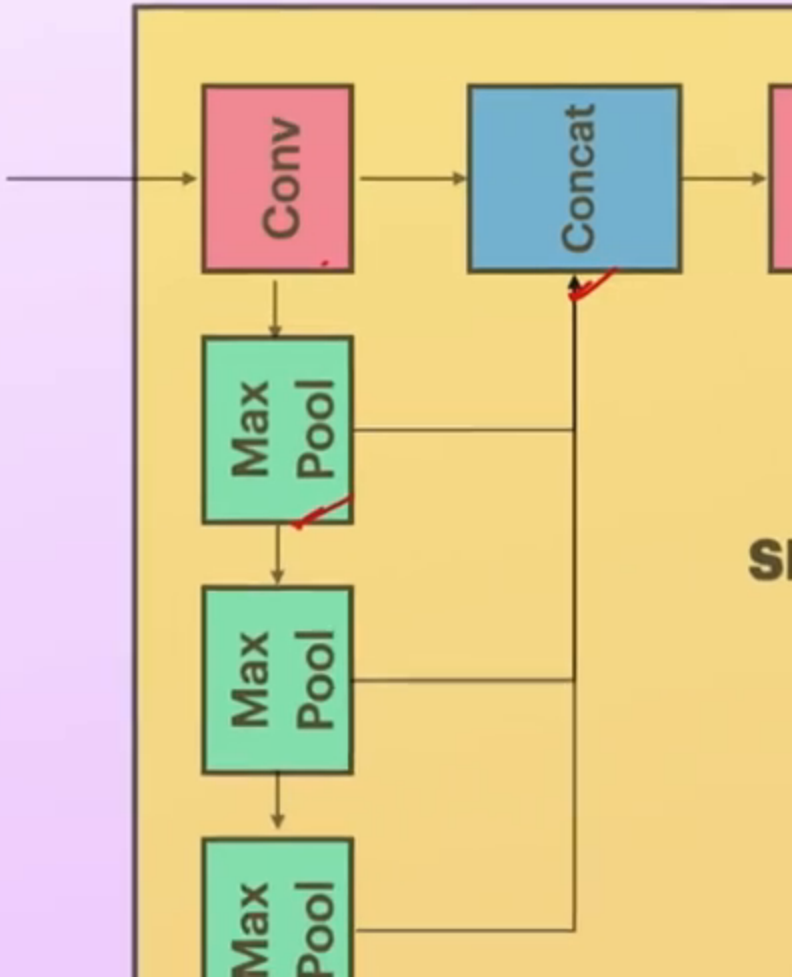


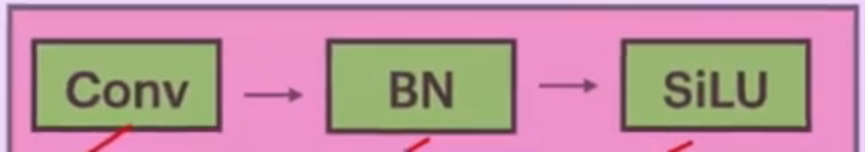


$$SiLU = x * sigmoid(x)$$

Backbone







$$SiLU = x * sigmoid(x)$$

Conv

Conv

C3

Conv

C3

Conv

N

*

Conv

Conv

Conv

Concat

Conv

Conv

C3

Conv

Conv

C3

Conv

C3

Conv

n

*

Conv

Conv

Conv

Concat

Conv

Conv

C3

Conv

Conv

C3

Conv

C3

Conv

N
*

Conv

Conv

Conv

Concat

Conv

Conv

C3

Conv

Conv

C3

Conv

C3

Conv

N
*

Conv

Conv

Conv

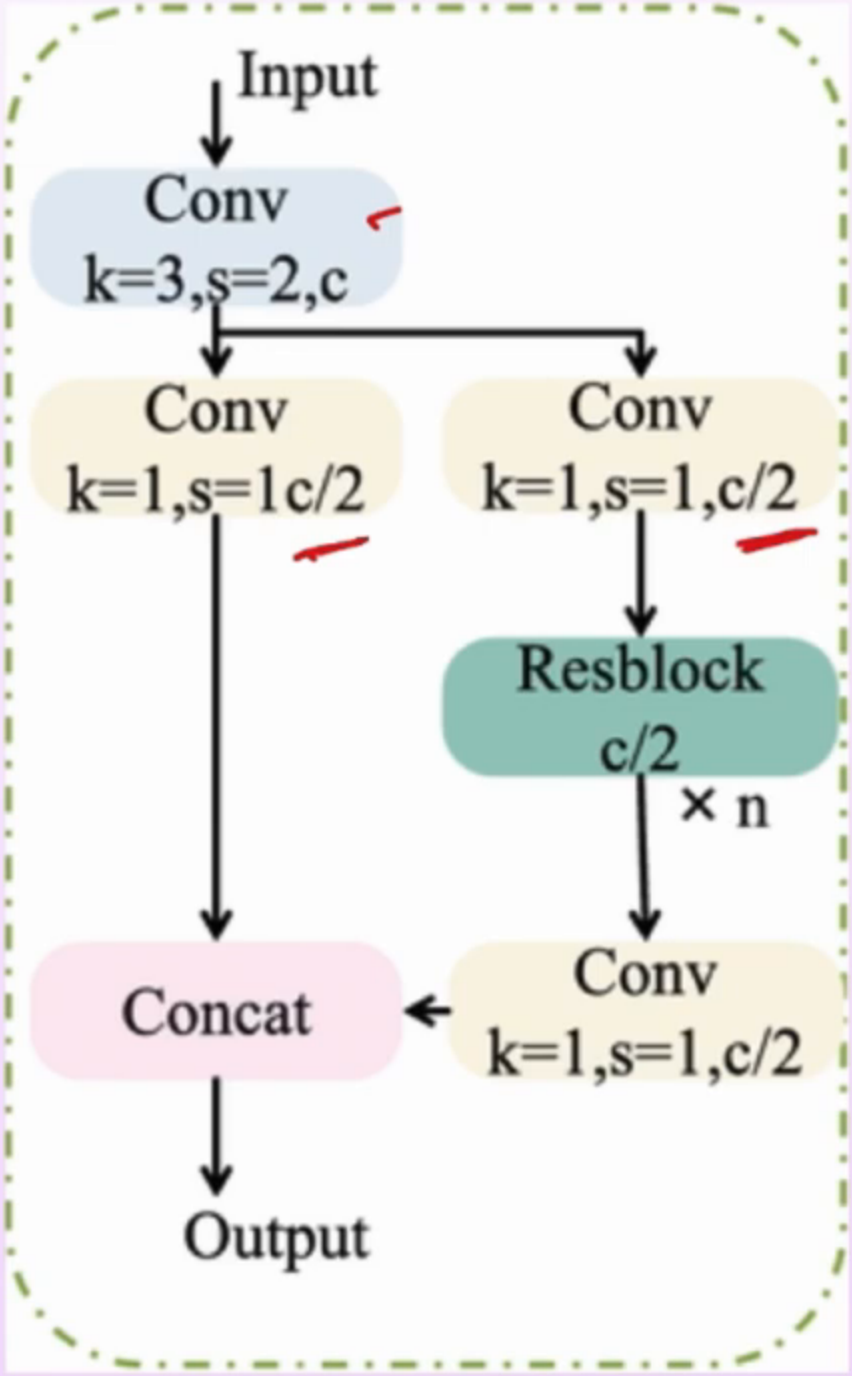
Concat

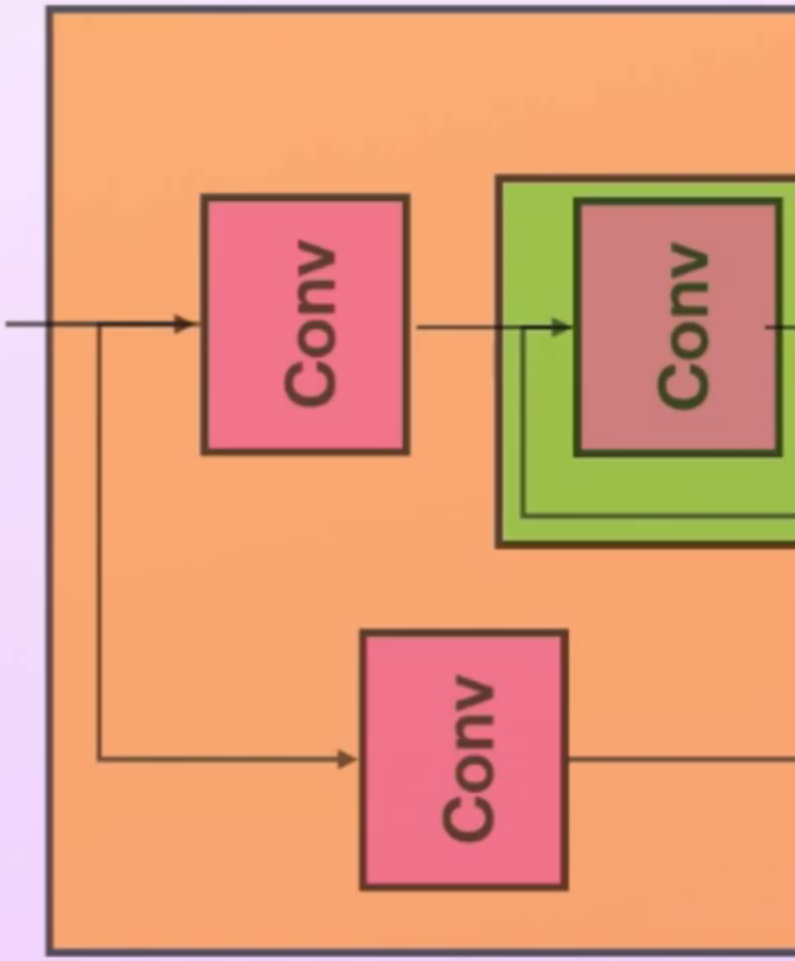
Conv

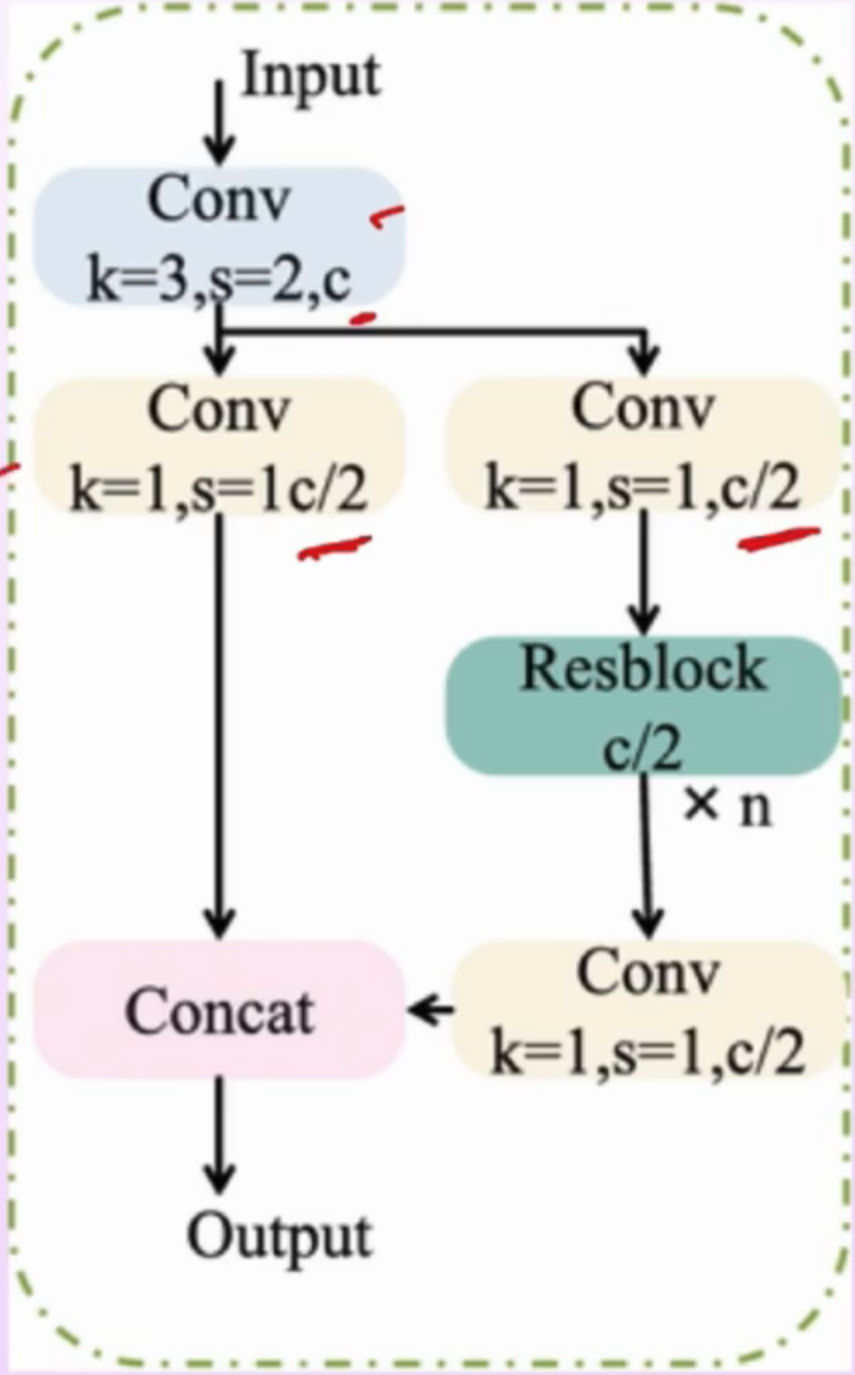
Conv

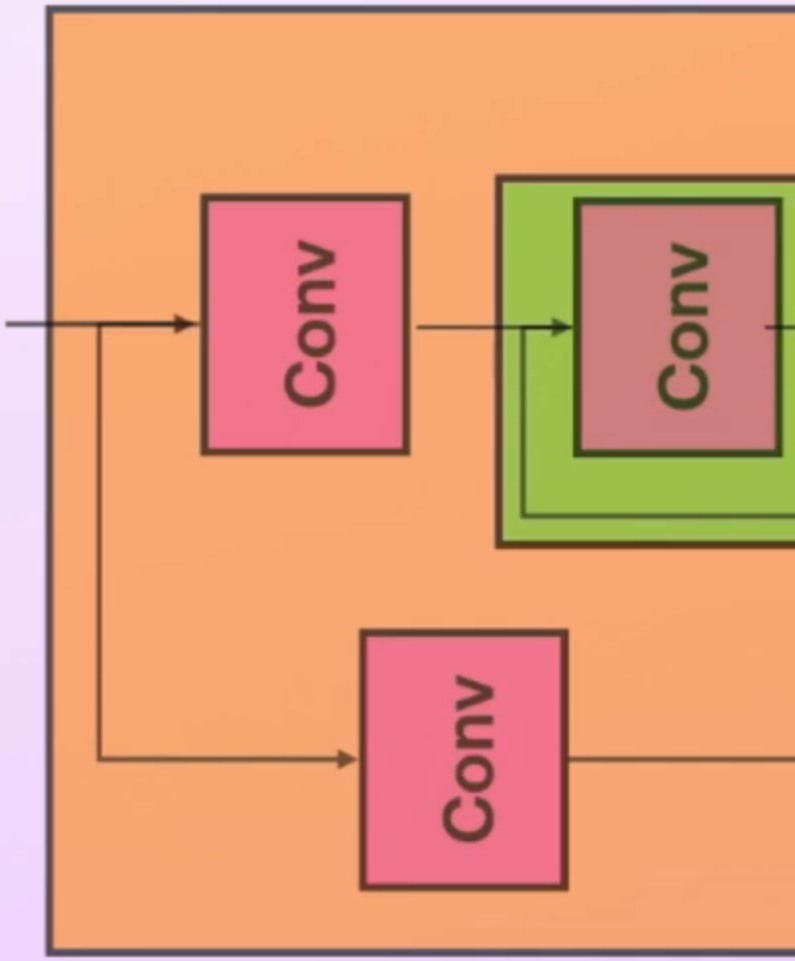
C3

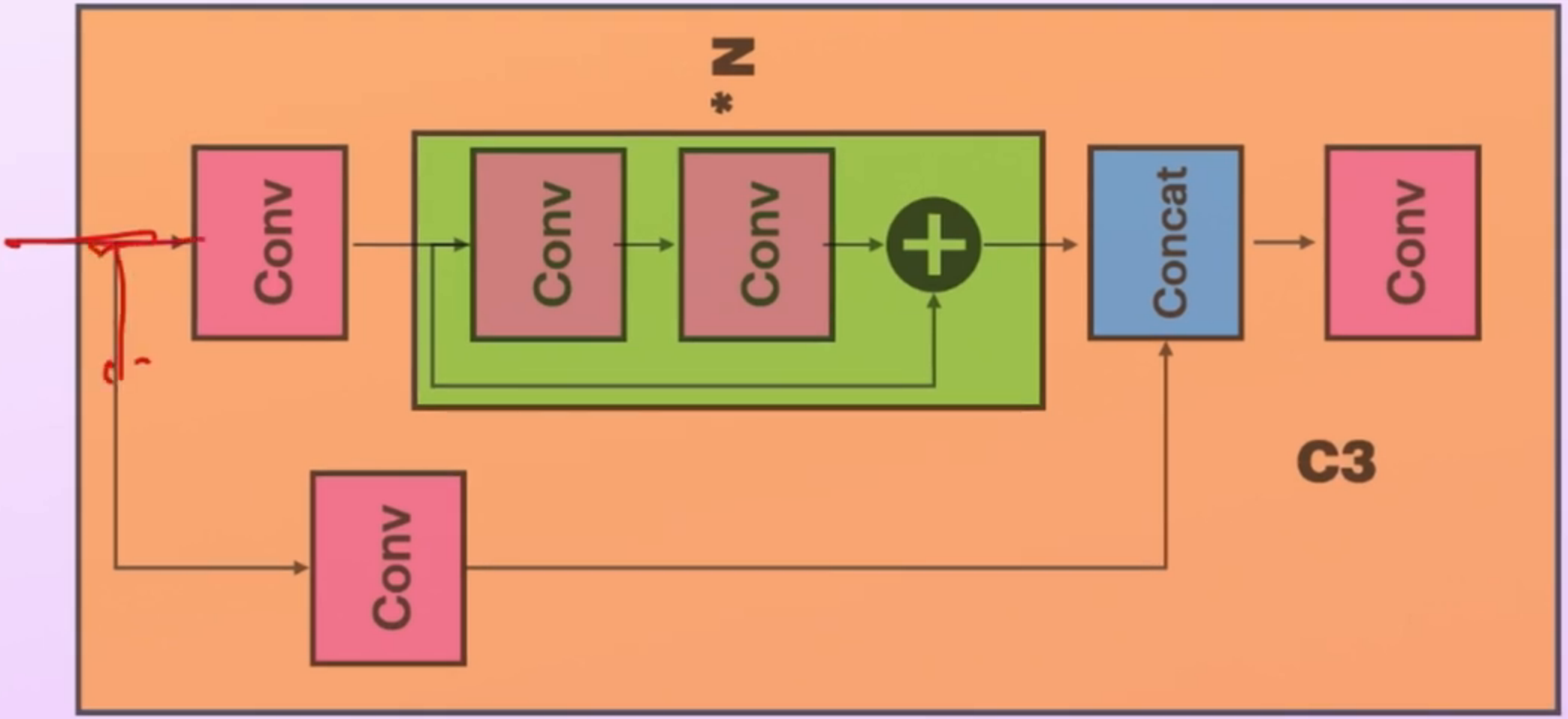
C3 Block


Input
Conv
k=3,s=2,c
Conv
k=1,s=1c/2
Conv
k=1,s=1,c/2
Resblock
c/2
× n
Concat
Conv
k=1,s=1,c/2
Output

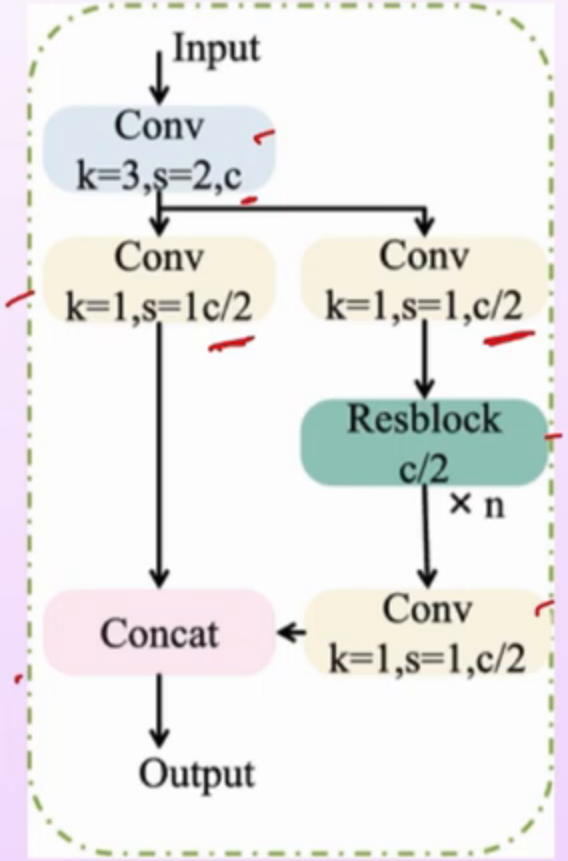

Conv
Conv
Conv

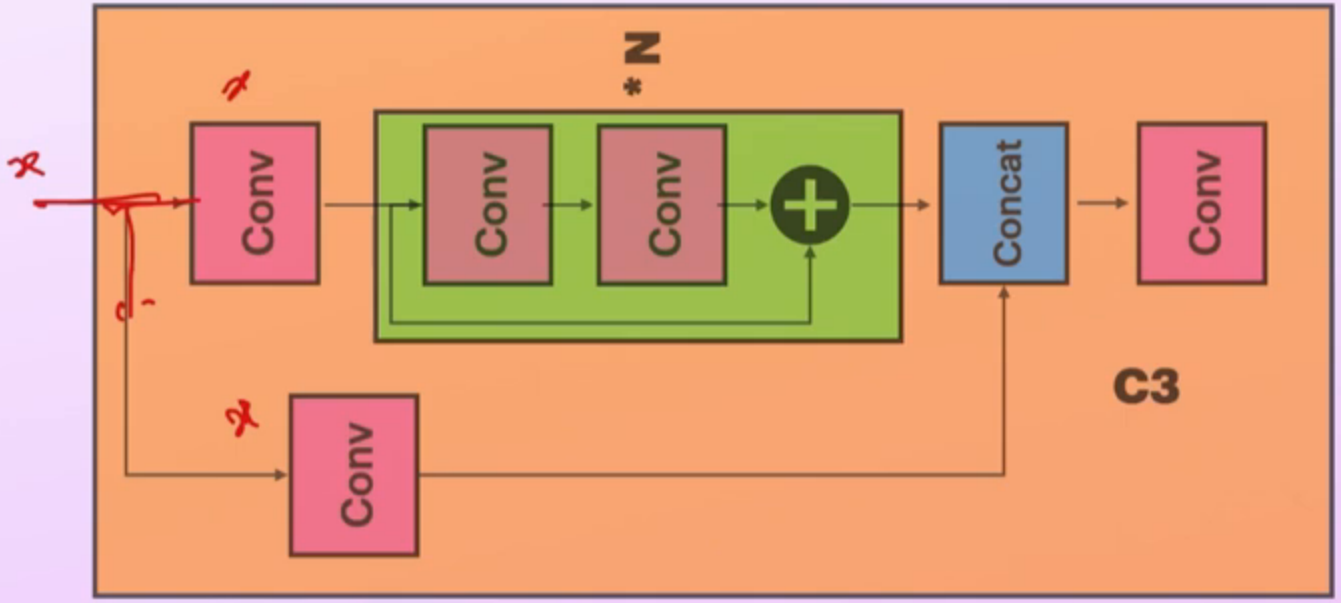

Input
Conv
k=3,s=2,c
Conv
k=1,s=1c/2
Conv
k=1,s=1,c/2
Resblock
c/2
× n
Concat
Conv
k=1,s=1,c/2
Output


Conv
Conv
Conv

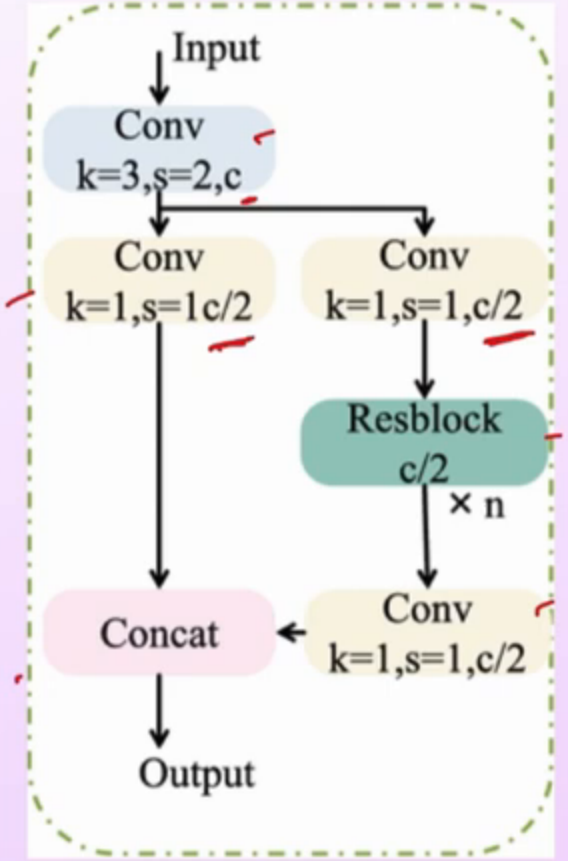

N
*
Conv
Conv
Conv
Concat
Conv
Conv
C3

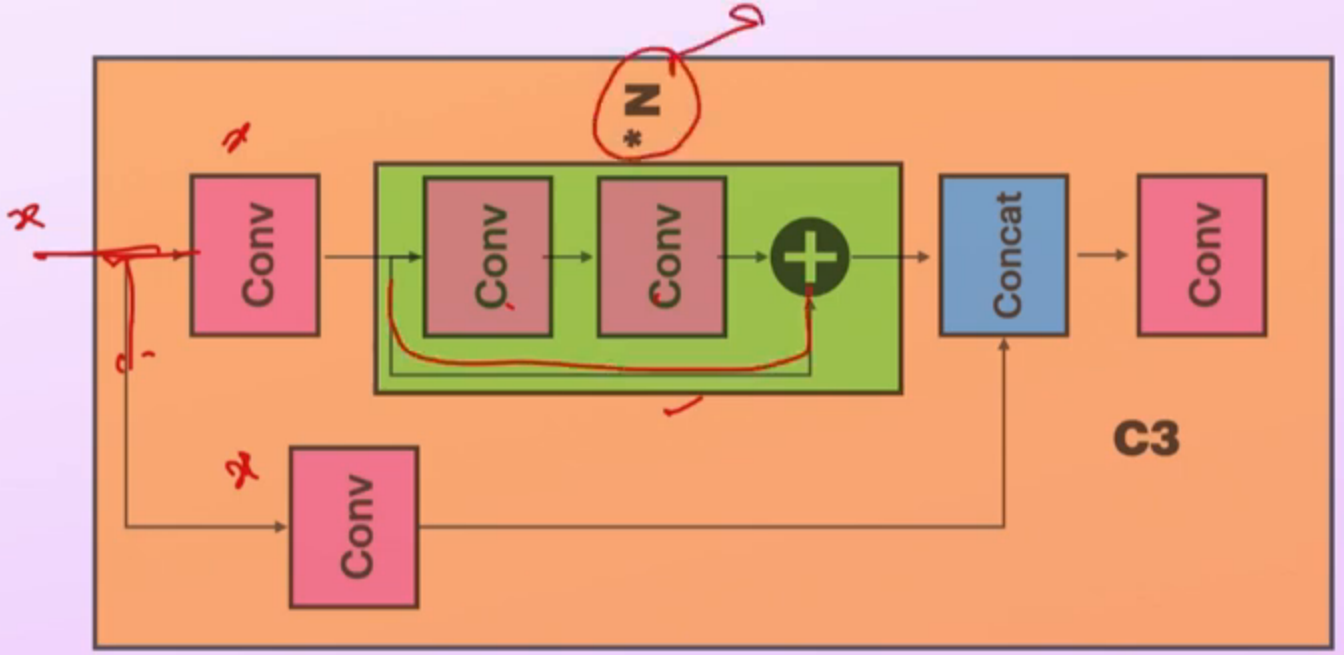
C3 Block


Input
Conv
k=3,s=2,c
Conv
k=1,s=1c/2
Conv
k=1,s=1,c/2
Resblock
c/2
× n
Concat
Conv
k=1,s=1,c/2
Output

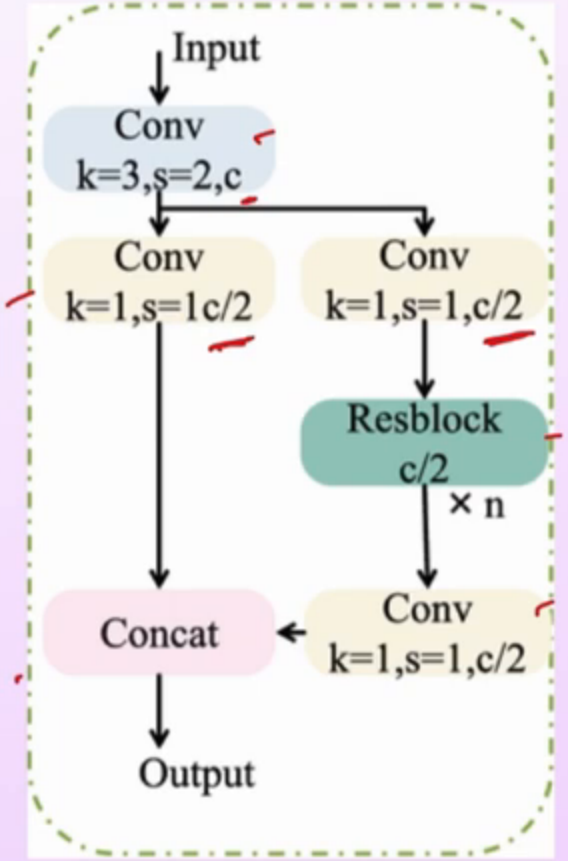

* N
Conv
Conv
Conv
Concat
Conv
Conv
C3

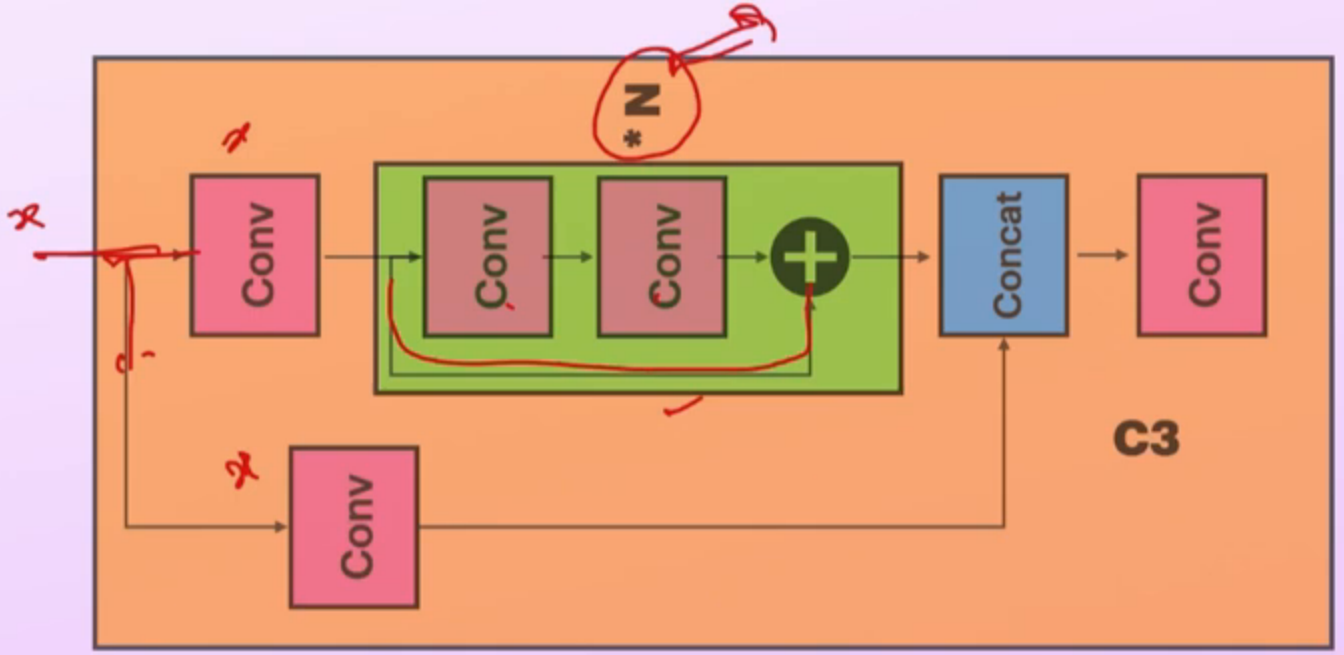
C3 Block


Input
Conv
k=3,s=2,c
Conv
k=1,s=1c/2
Conv
k=1,s=1,c/2
Resblock
c/2
× n
Concat
Conv
k=1,s=1,c/2
Output

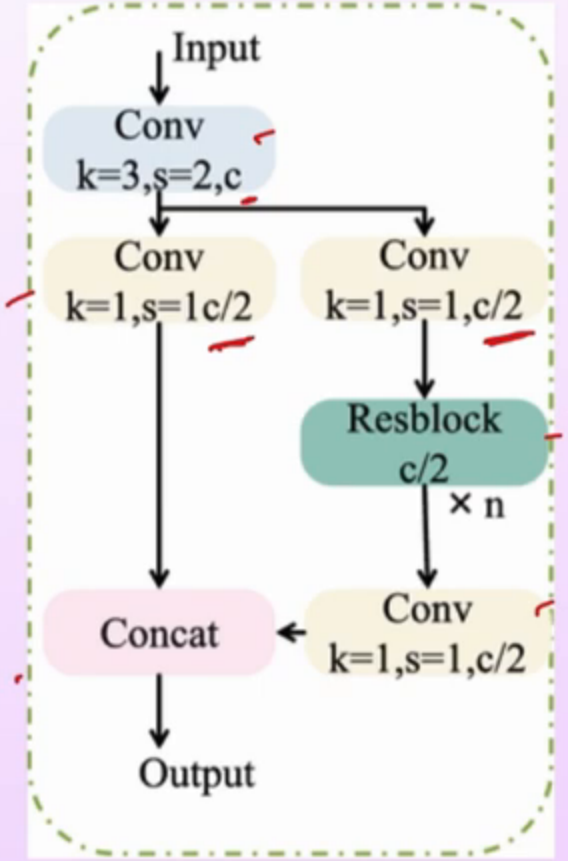

N
*
Conv
Conv
Conv
Concat
Conv
Conv
C3

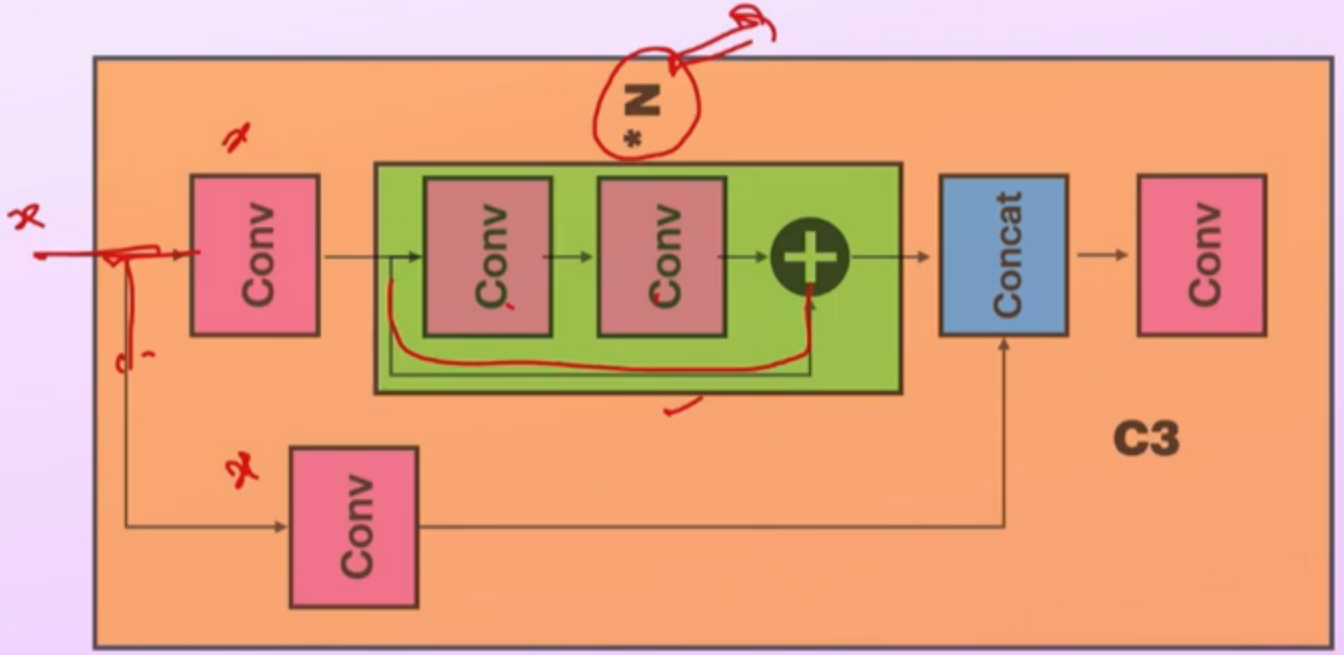
C3 Block


Input
Conv
k=3,s=2,c
Conv
k=1,s=1c/2
Conv
k=1,s=1,c/2
Resblock
c/2
× n
Concat
Conv
k=1,s=1,c/2
Output

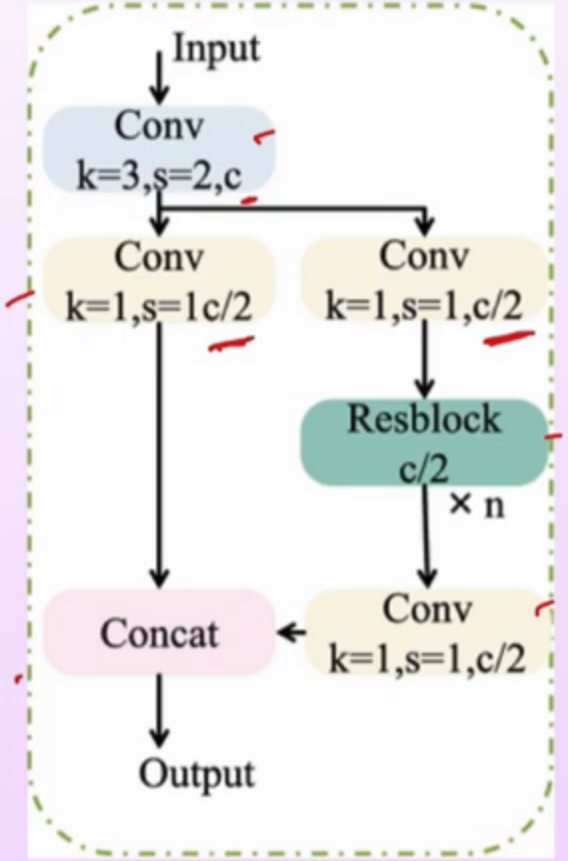

* N
Conv
Conv
Conv
Concat
Conv
Conv
C3

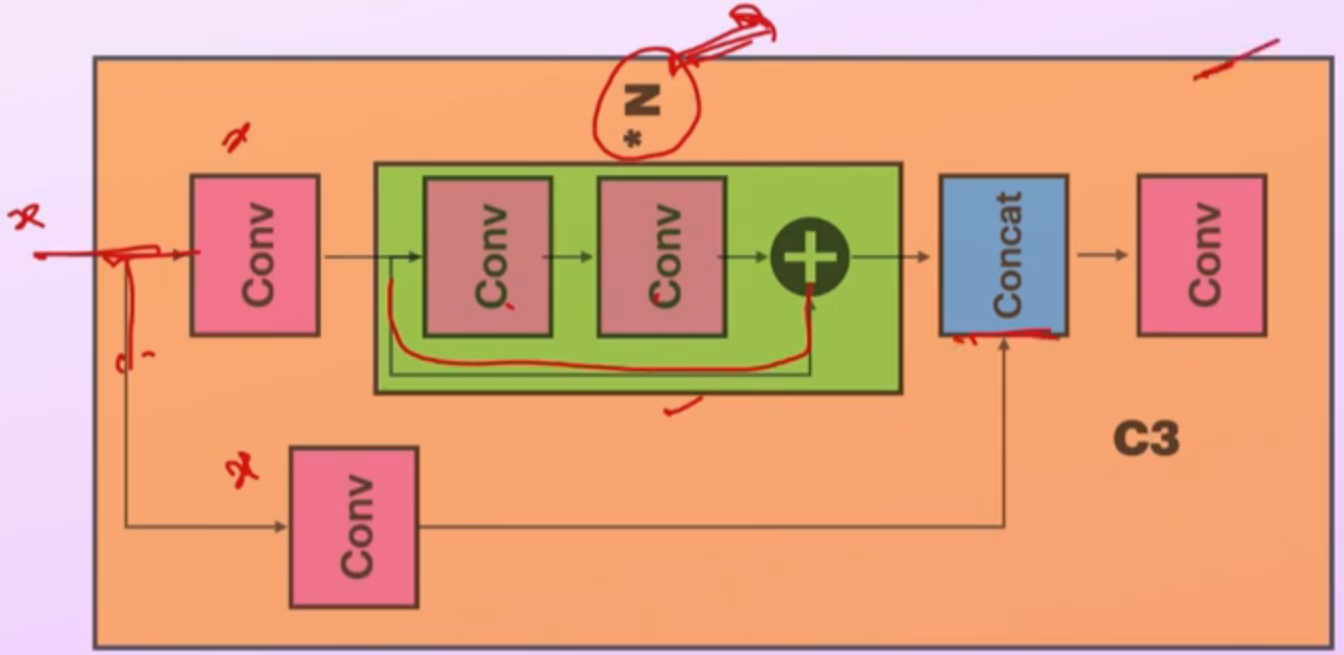
C3 Block


Input
Conv
k=3,s=2,c
Conv
k=1,s=1c/2
Conv
k=1,s=1,c/2
Resblock
c/2
× n
Concat
Conv
k=1,s=1,c/2
Output

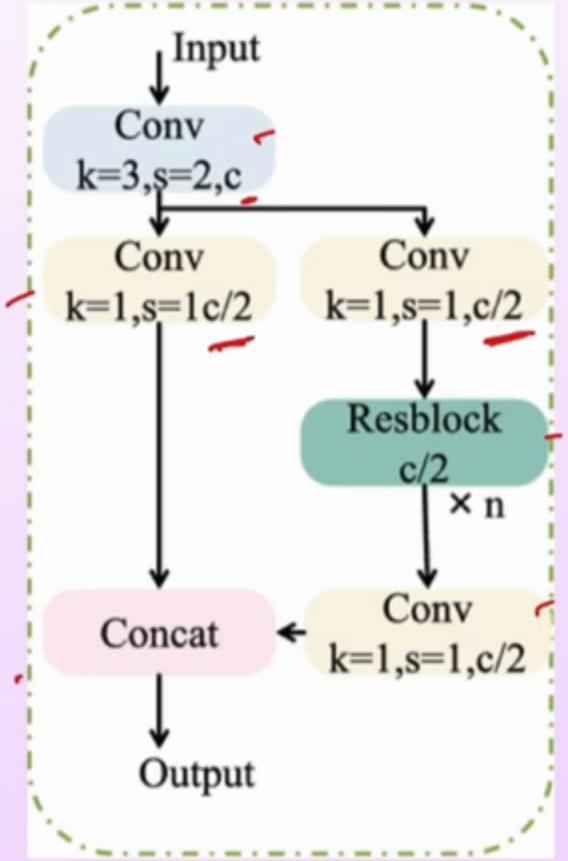

* N
Conv
Conv
Conv
Concat
Conv
Conv
C3

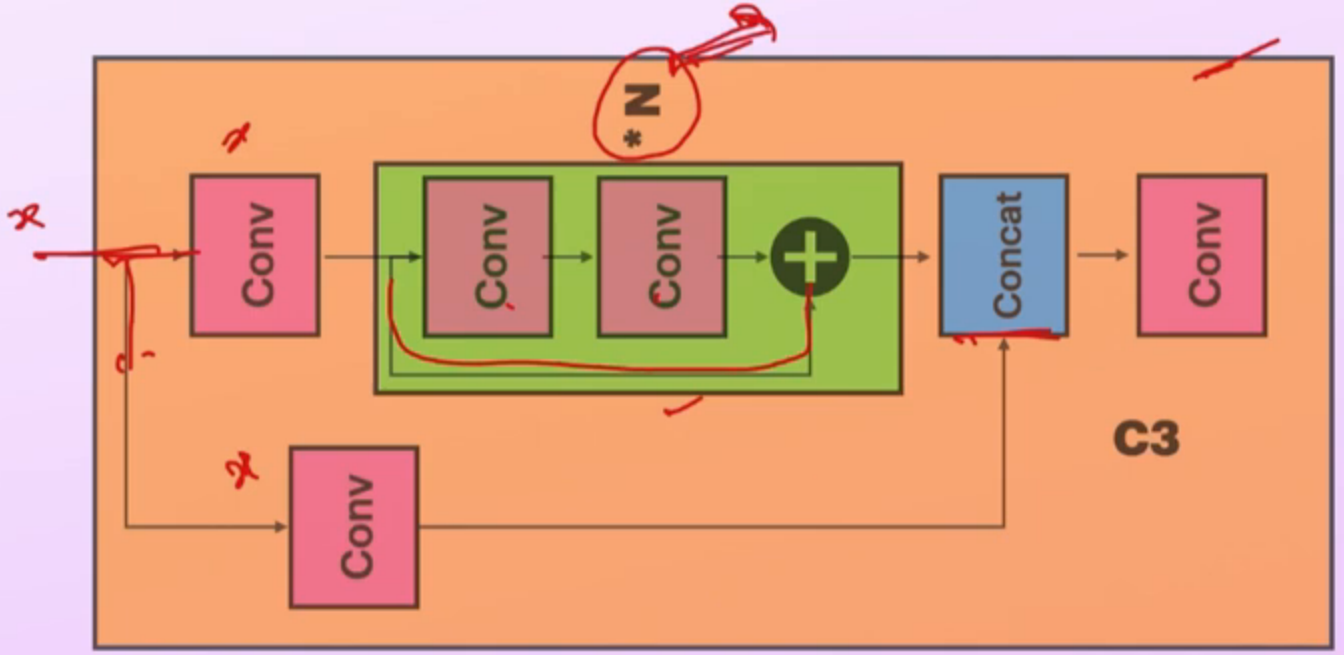
C3 Block


Input
Conv
k=3,s=2,c
Conv
k=1,s=1c/2
Conv
k=1,s=1,c/2
Resblock
c/2
× n
Concat
Conv
k=1,s=1,c/2
Output

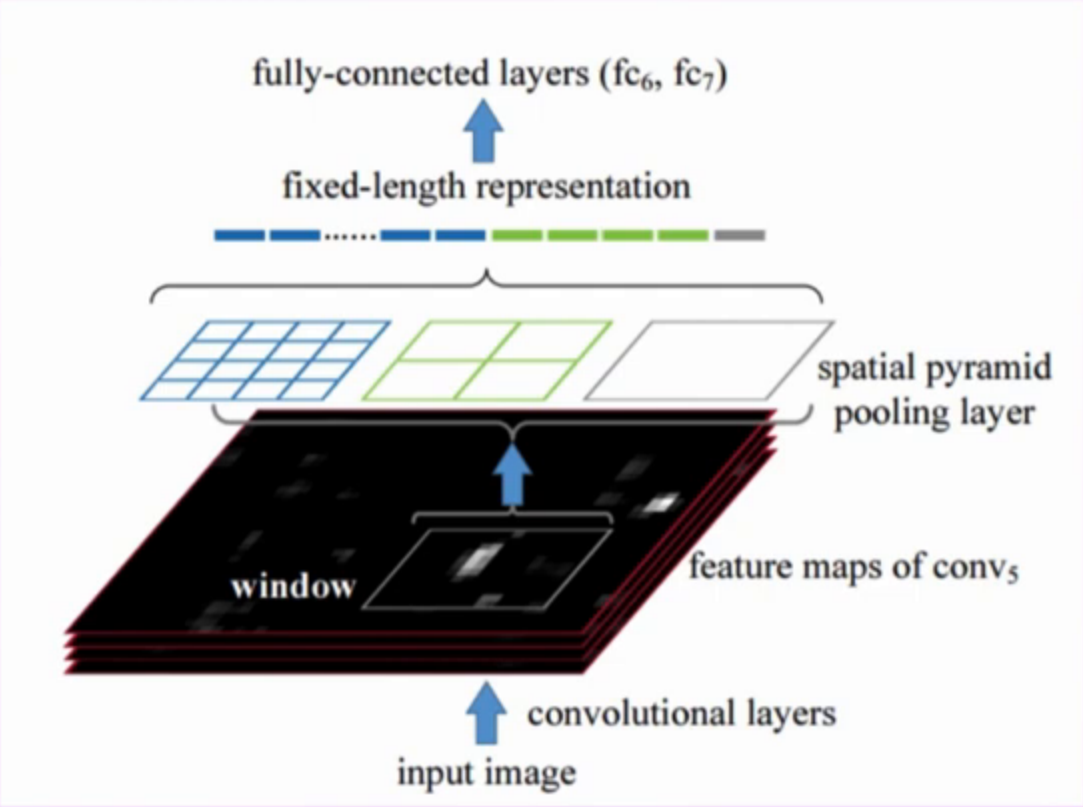

* N
Conv
Conv
Conv
Concat
Conv
Conv
C3

C3 Block

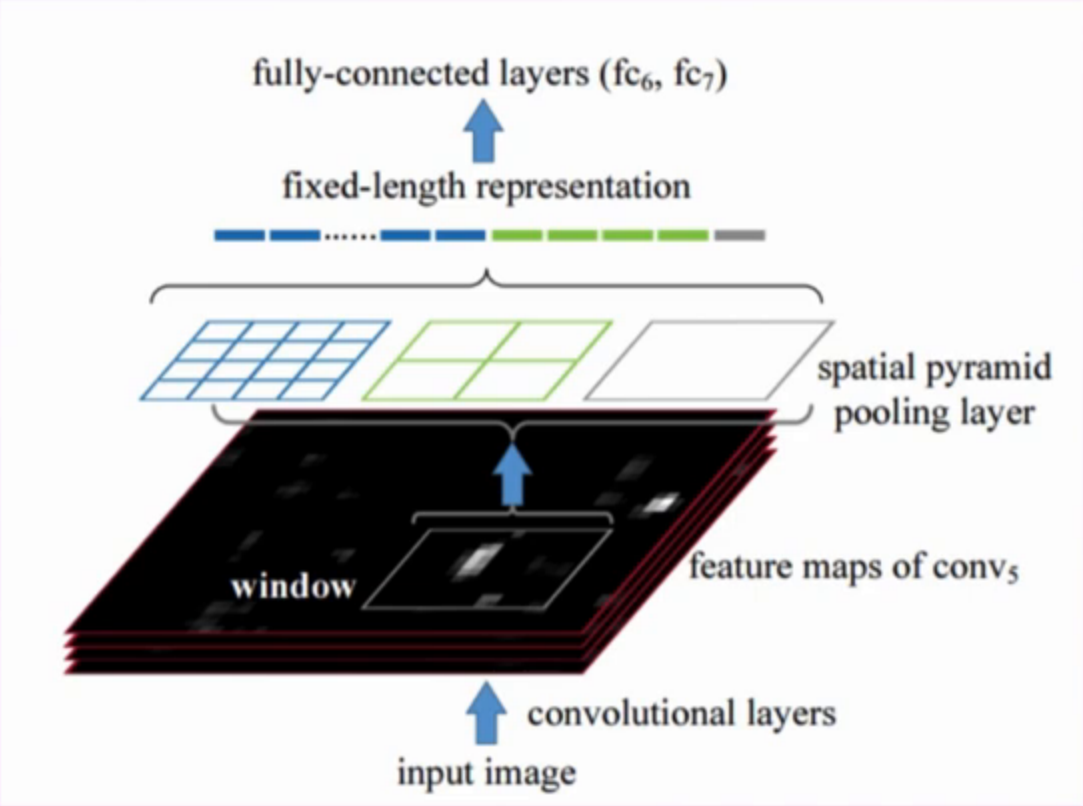

Input
Conv
k=3,s=2,c
Conv
k=1,s=1c/2
Conv
k=1,s=1,c/2
Resblock
c/2
× n
Concat
Conv
k=1,s=1,c/2
Output


* N
Conv
Conv
Conv
Concat
Conv
Conv
C3

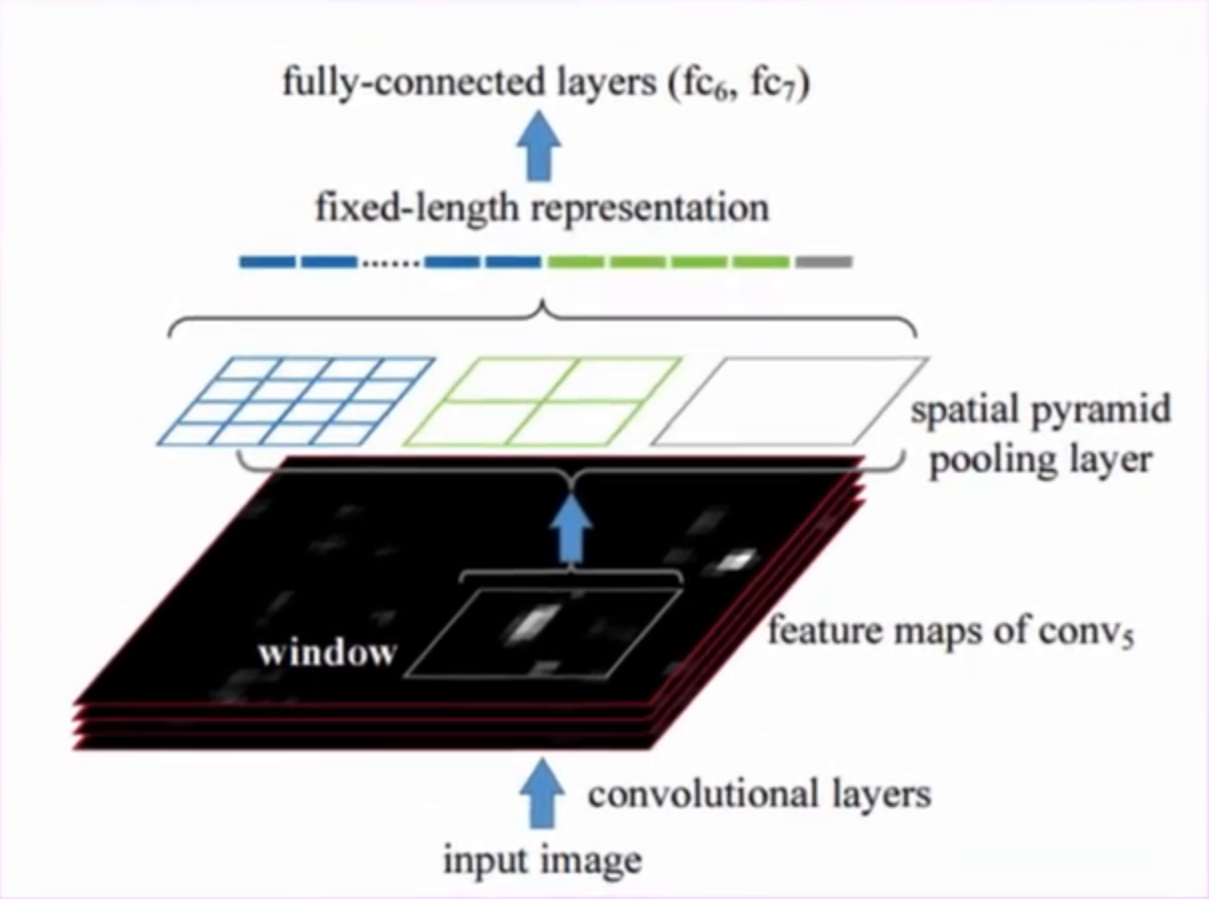
Spatial Pyramid Pooling Fast

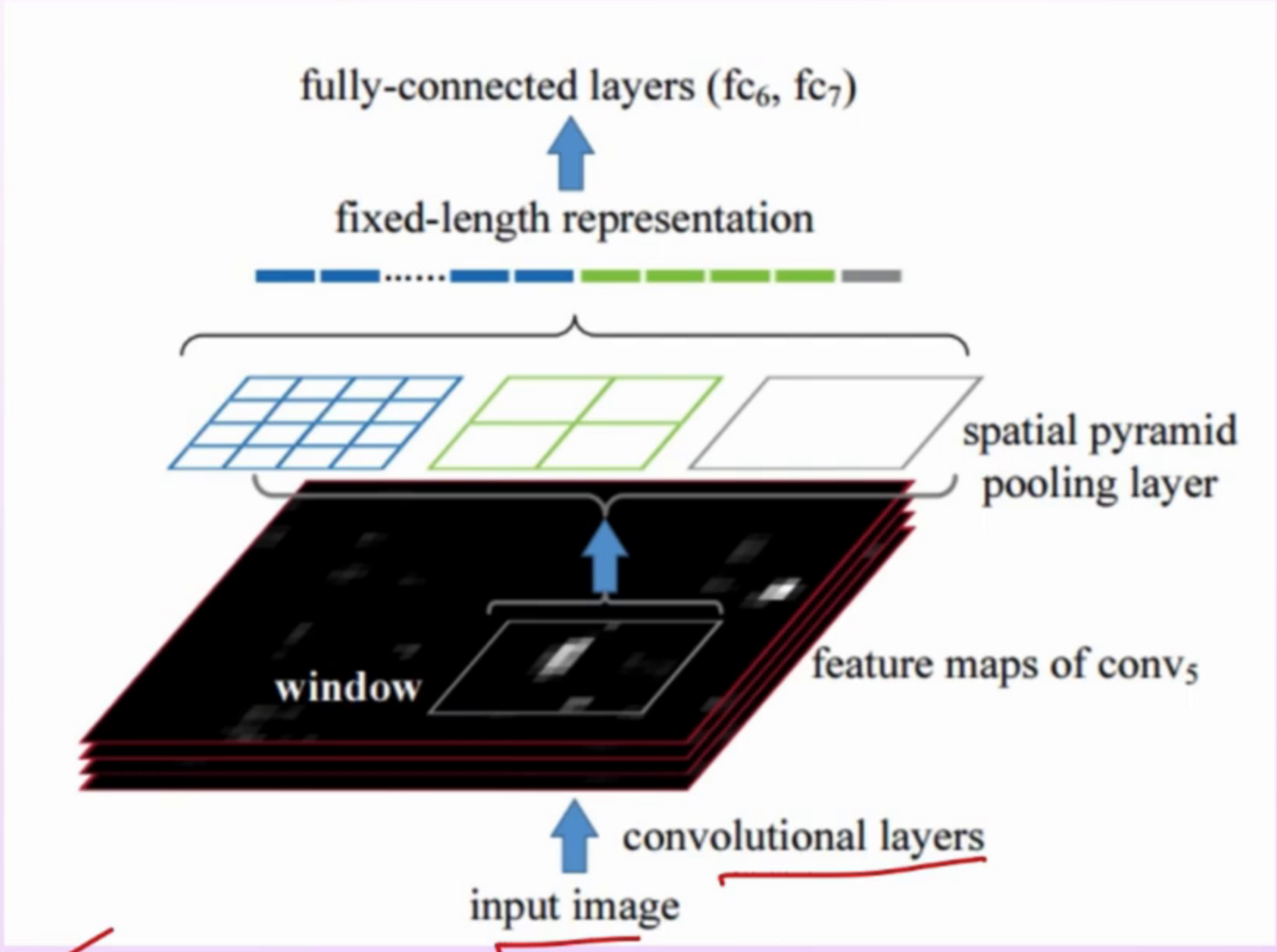


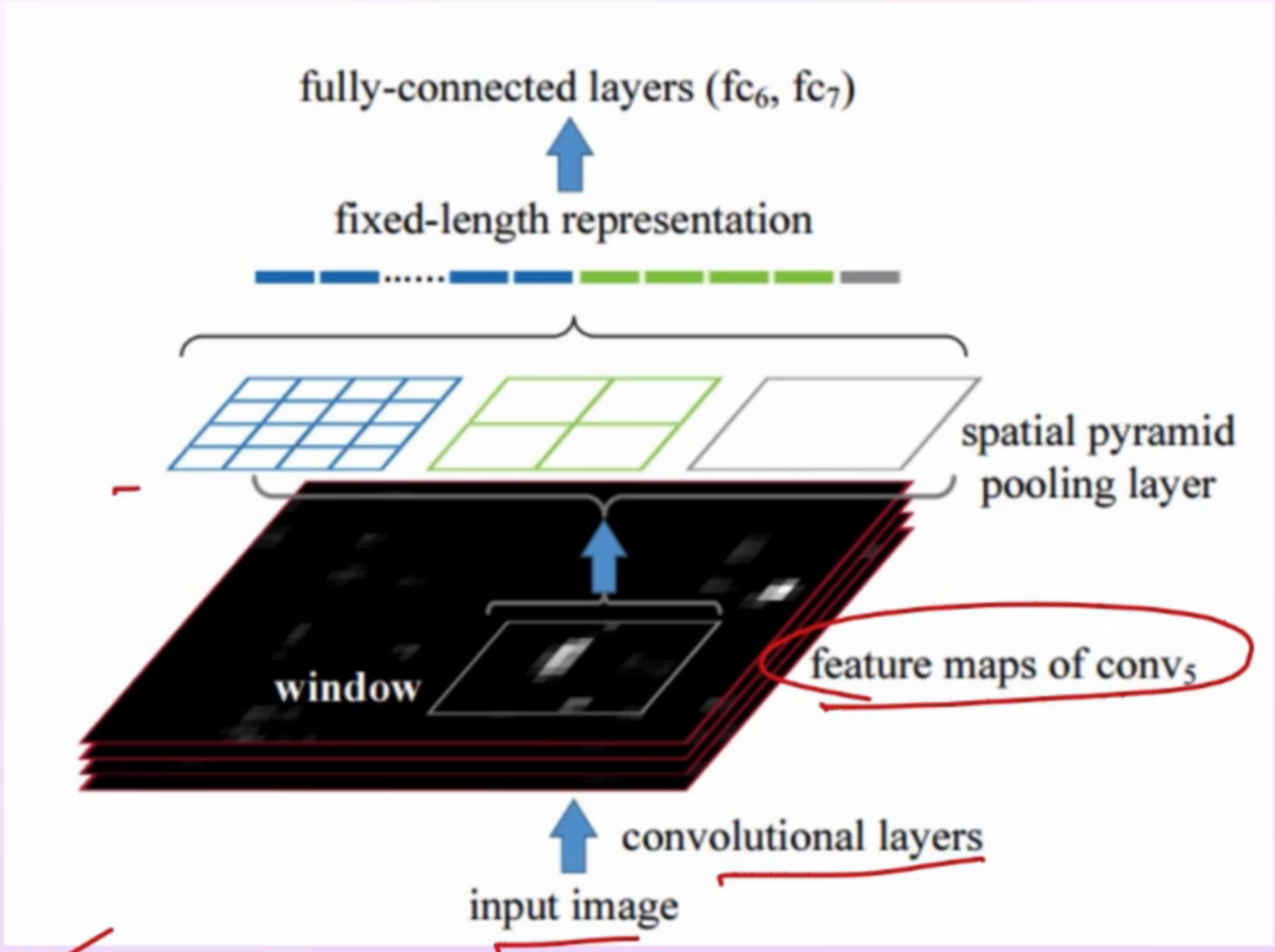
Spatial Pyramid Pooling Fast

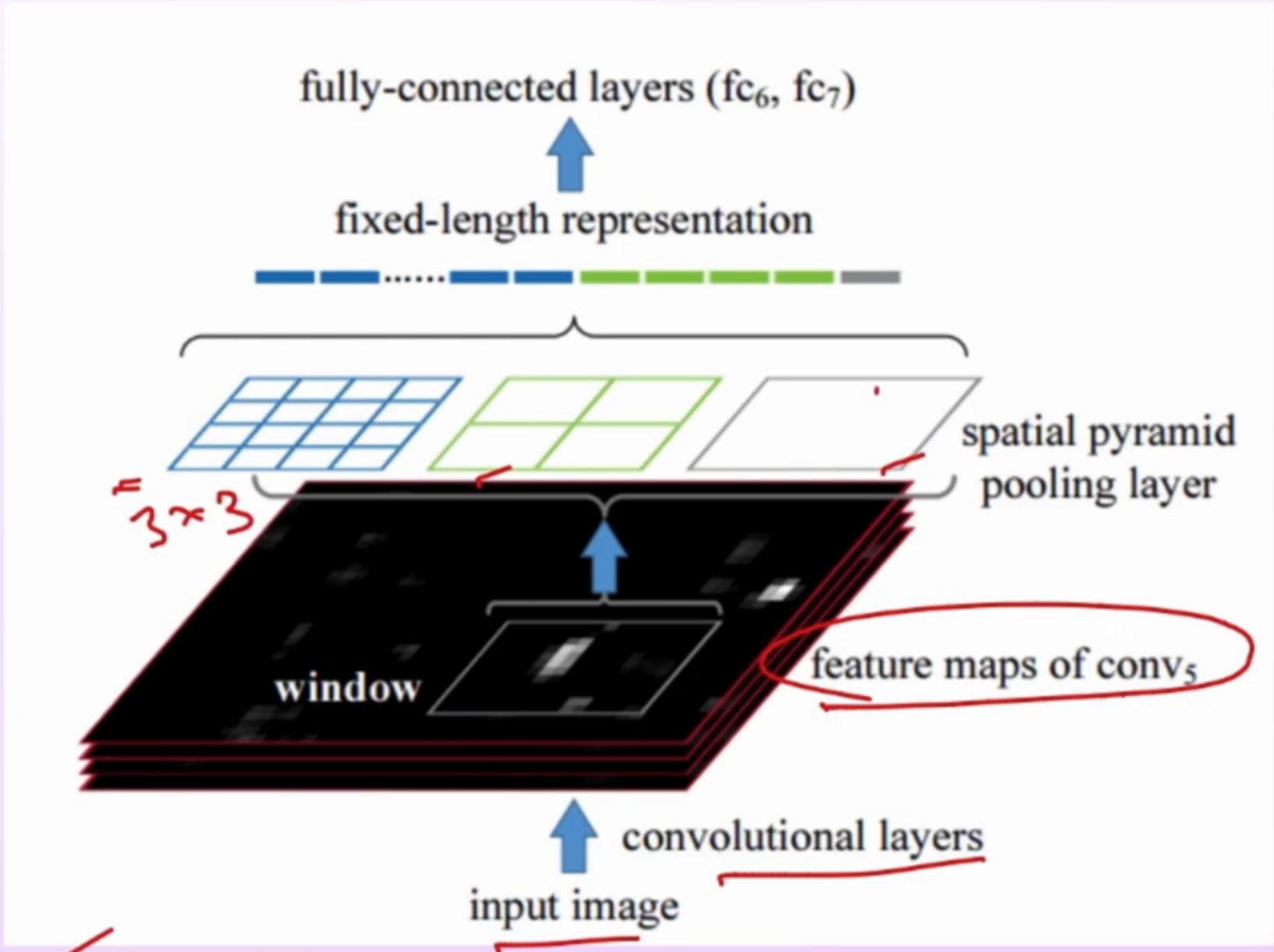


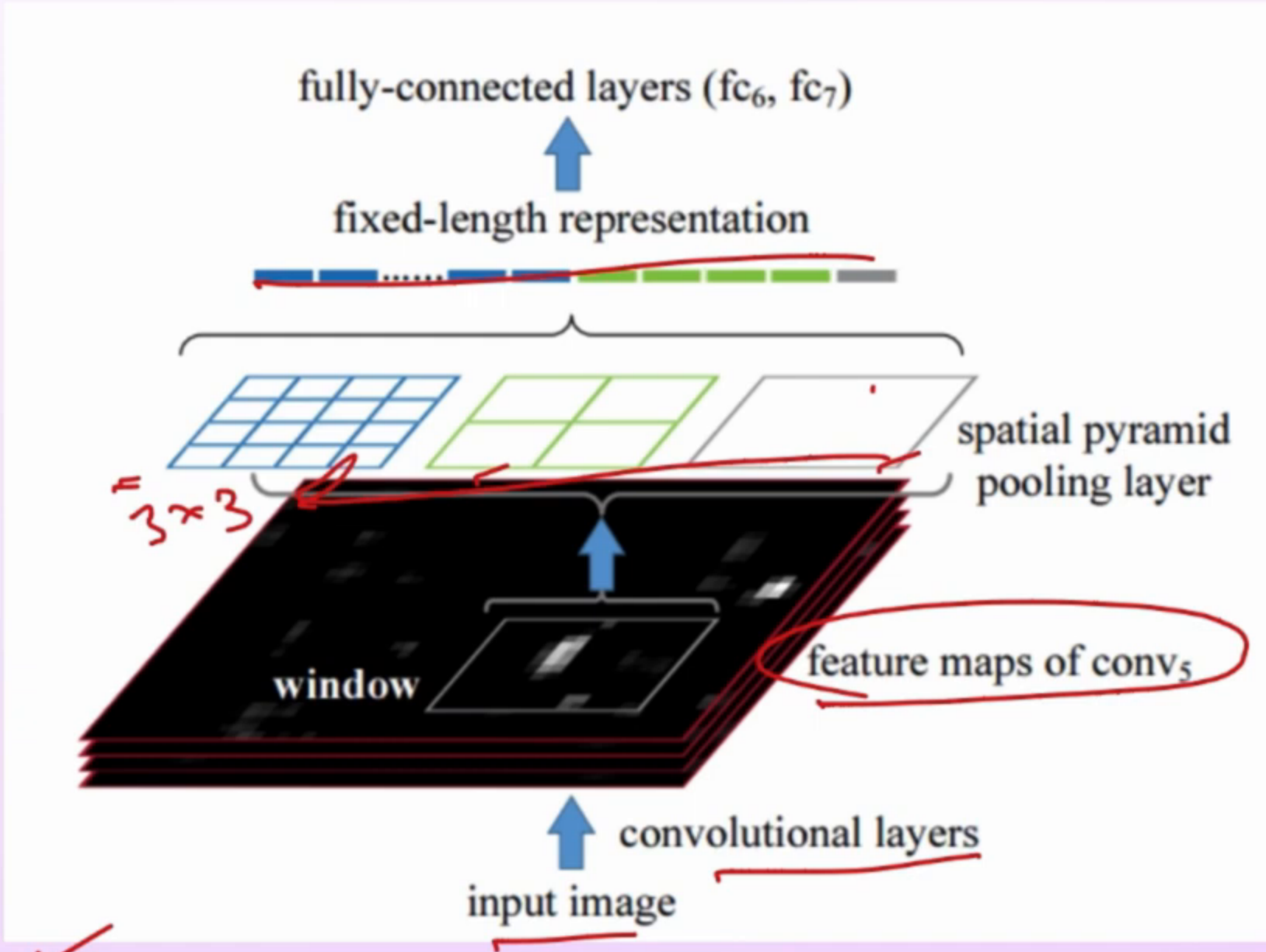
Spatial Pyramid Pooling Fast

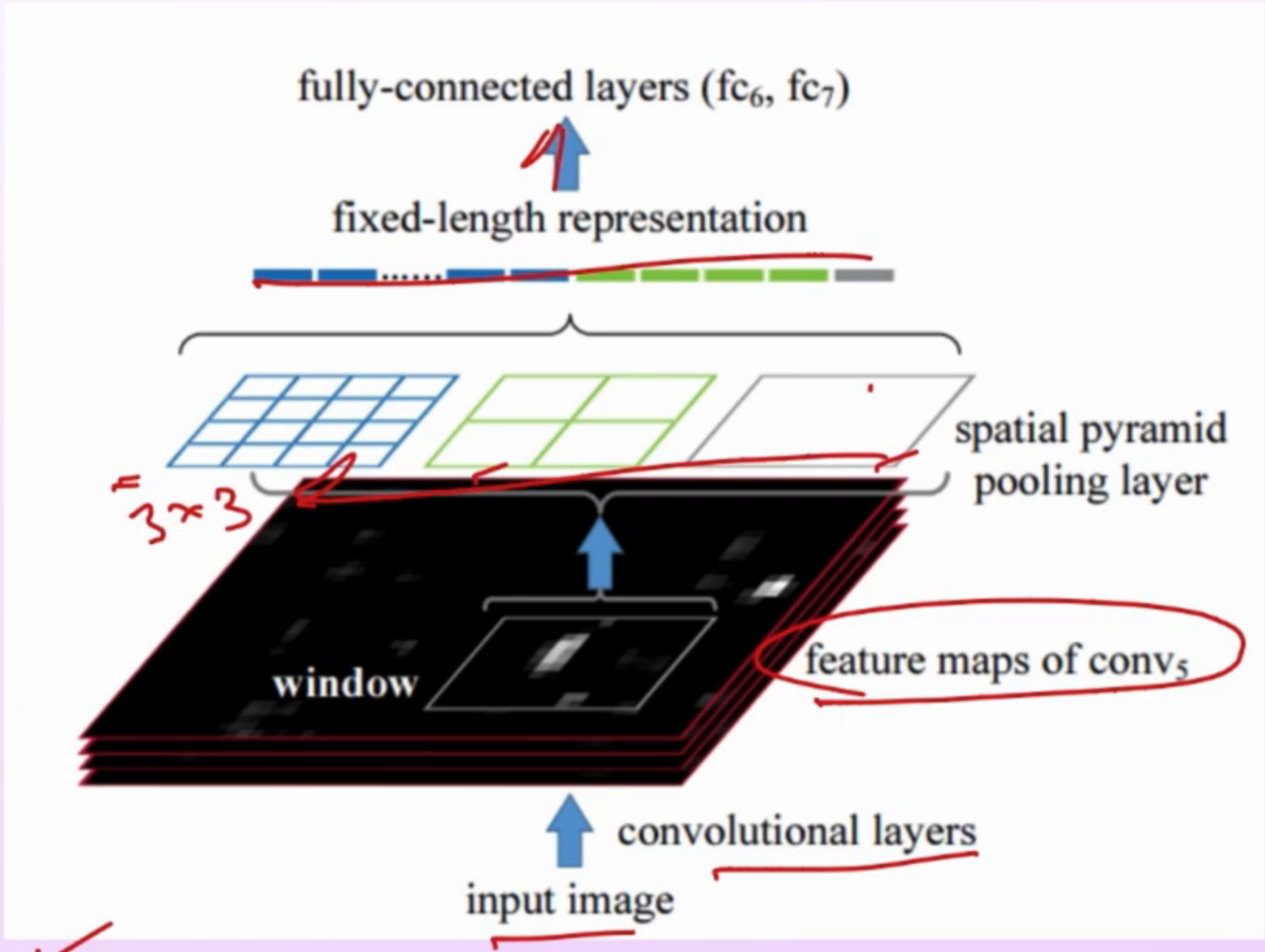


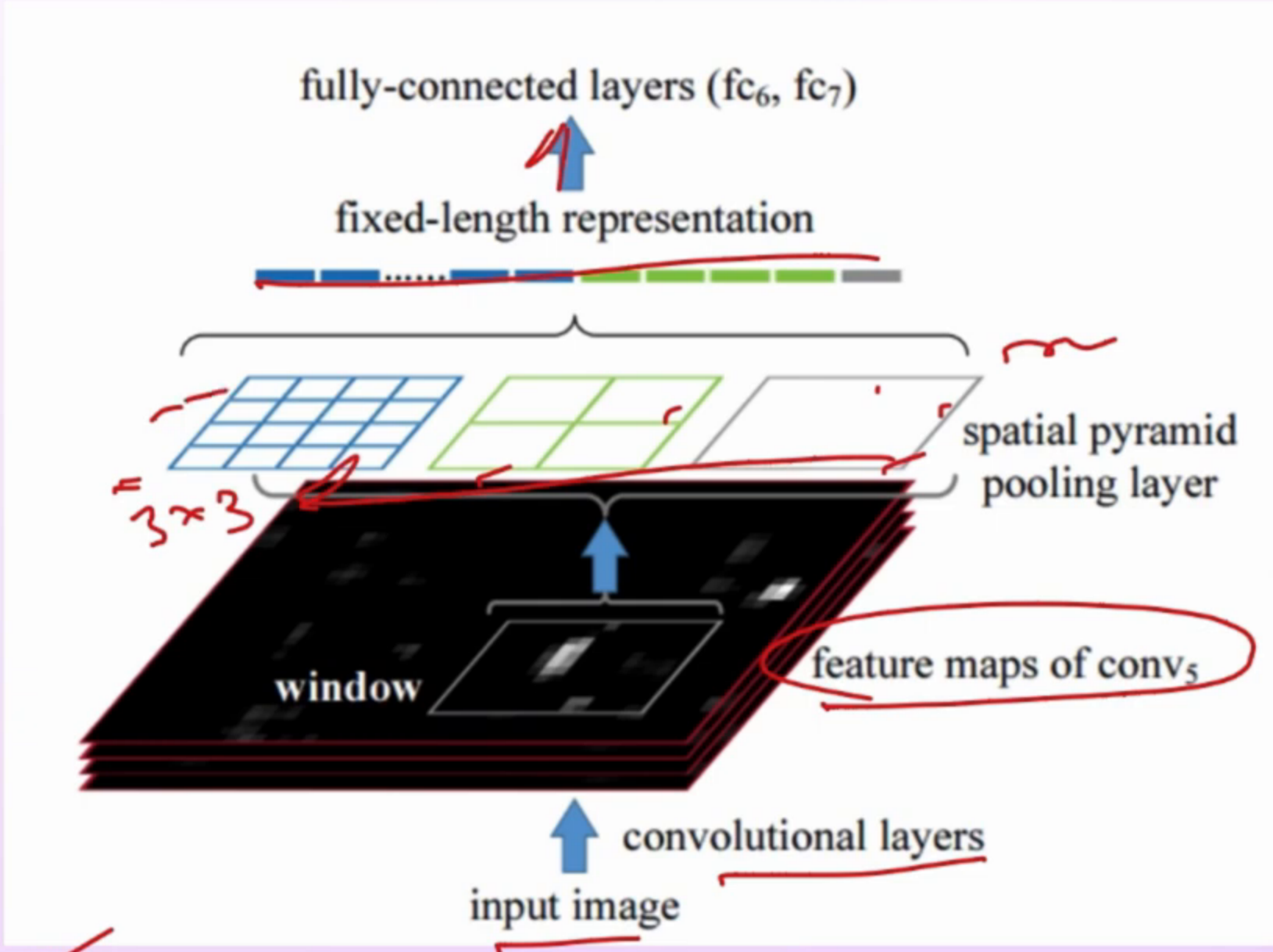

fully-connected layers (fc_6, fc_7)
fixed-length representation
spatial pyramid pooling layer
feature maps of $conv_5$
window
convolutional layers
input image

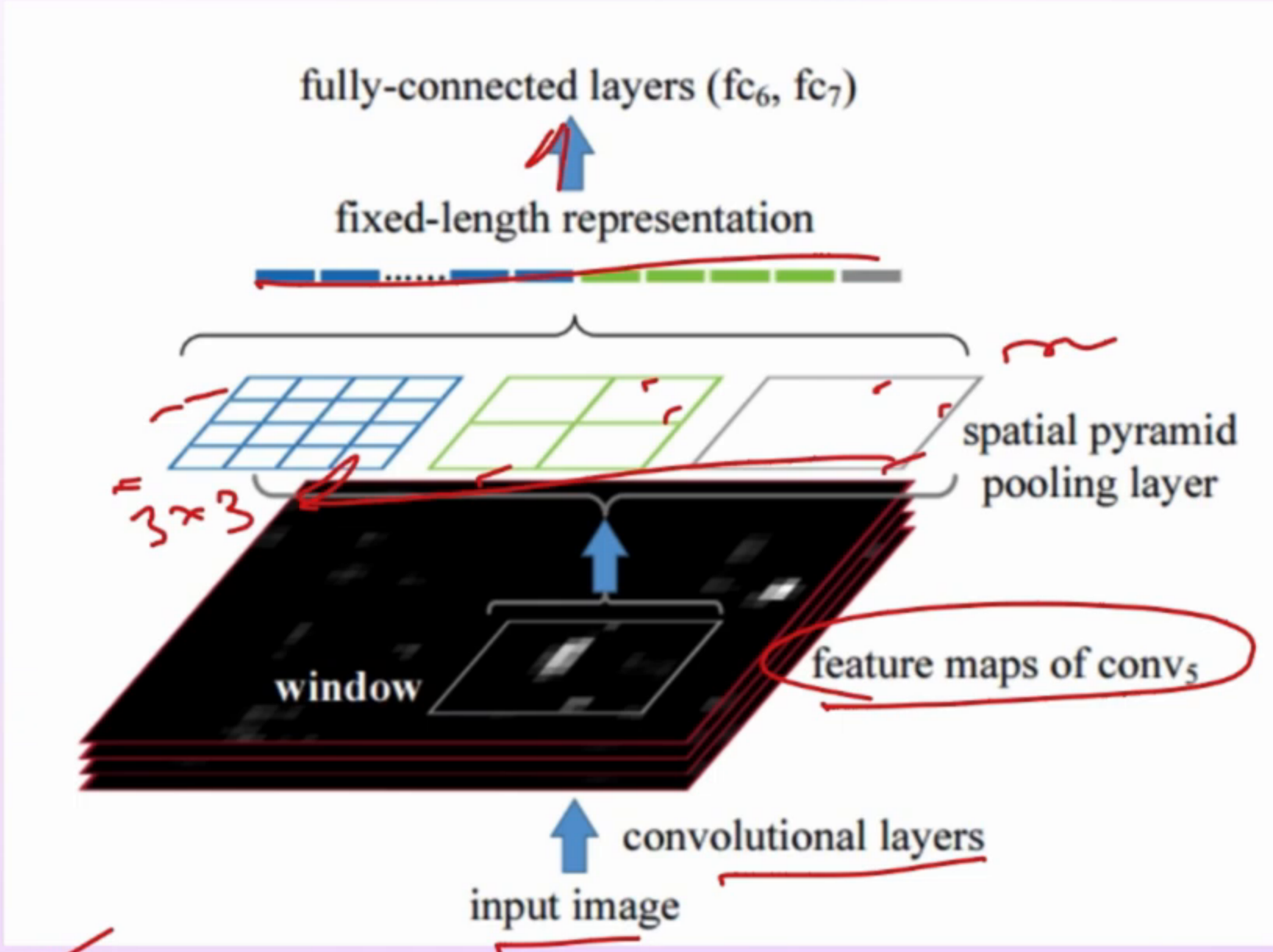

fully-connected layers (fc_6, fc_7)
fixed-length representation
spatial pyramid pooling layer
feature maps of $conv_5$
window
convolutional layers
input image

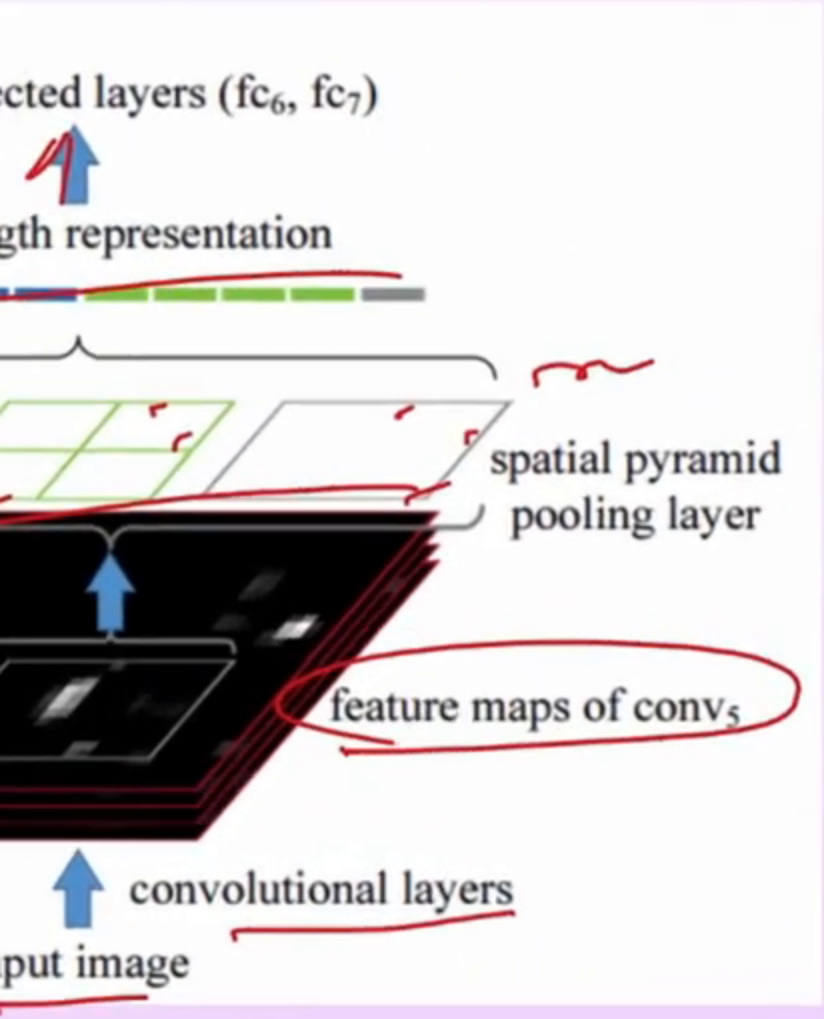

fully-connected layers (fc_6, fc_7)
fixed-length representation
spatial pyramid pooling layer
3x3
window
feature maps of $conv_5$
convolutional layers
input image

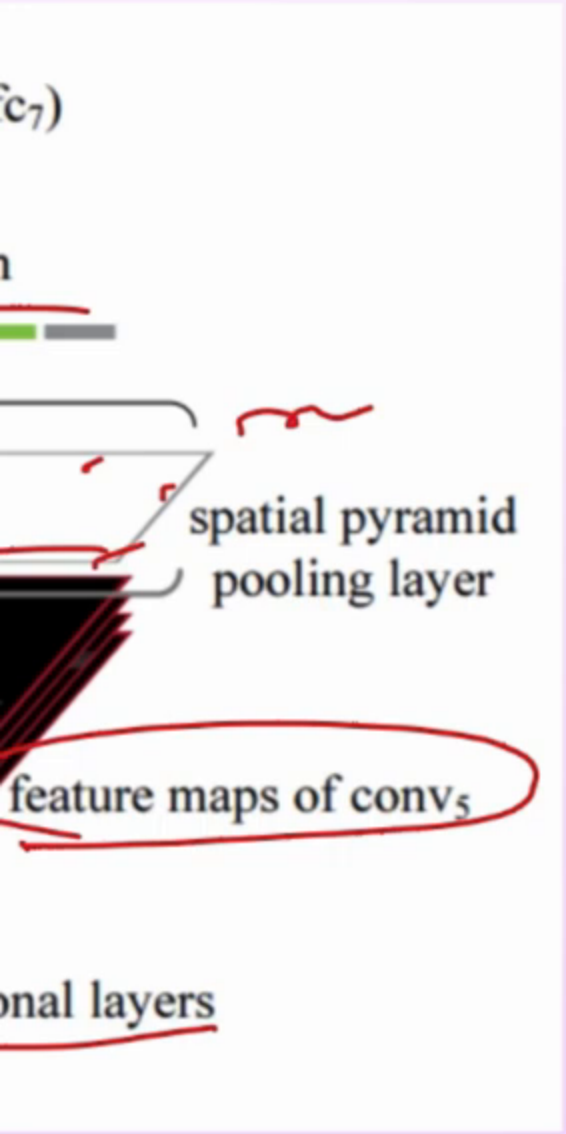

fully-connected layers (fc_6, fc_7)
fixed-length representation
spatial pyramid pooling layer
3×3
window
feature maps of $conv_5$
convolutional layers
input image

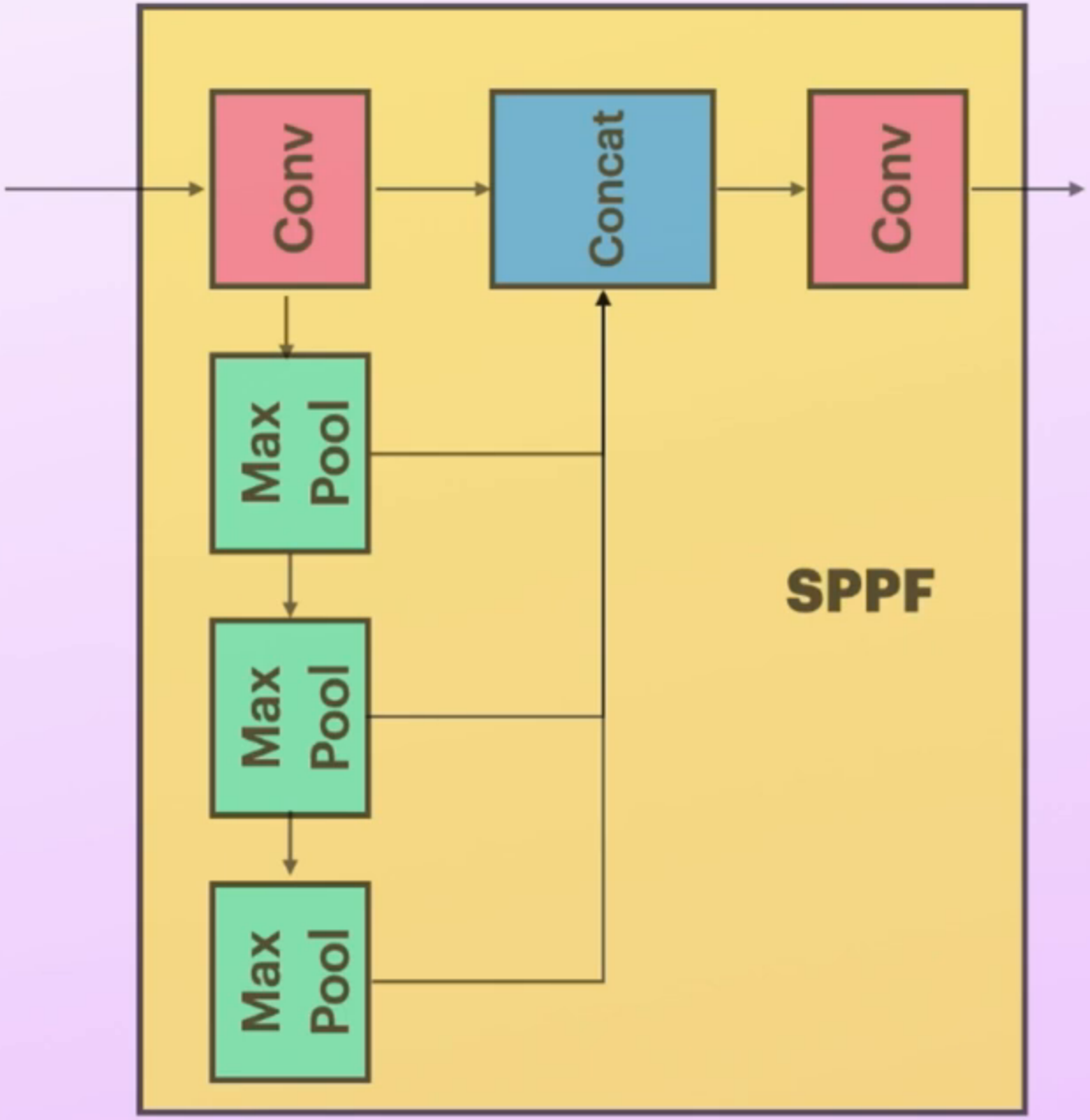

fully-connected layers (fc_6, fc_7)
fixed-length representation
spatial pyramid pooling layer
3x3
window
feature maps of $conv_5$
convolutional layers
input image


fully-connected layers (fc_6, fc_7)
fixed-length representation
spatial pyramid pooling layer
window
feature maps of $conv_5$
convolutional layers
input image

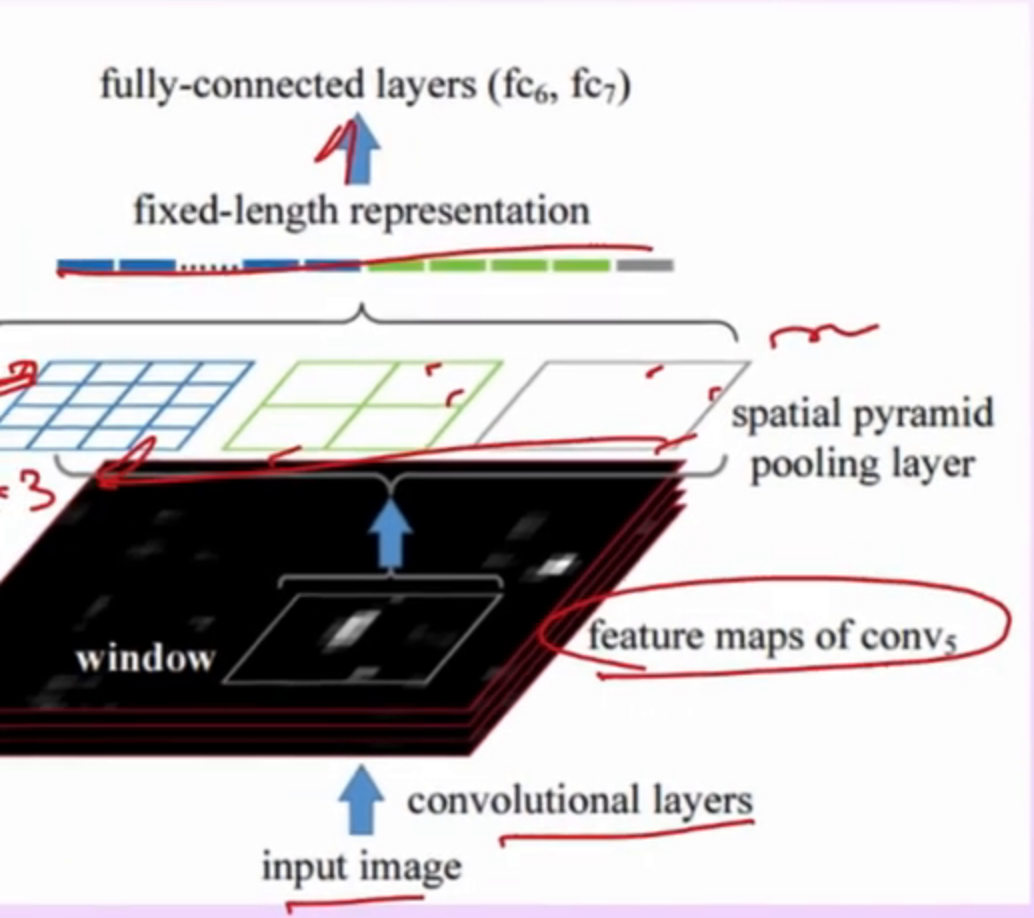

fully-connected layers (fc_6, fc_7)
fixed-length representation
spatial pyramid pooling layer
window
feature maps of $conv_5$
convolutional layers
input image

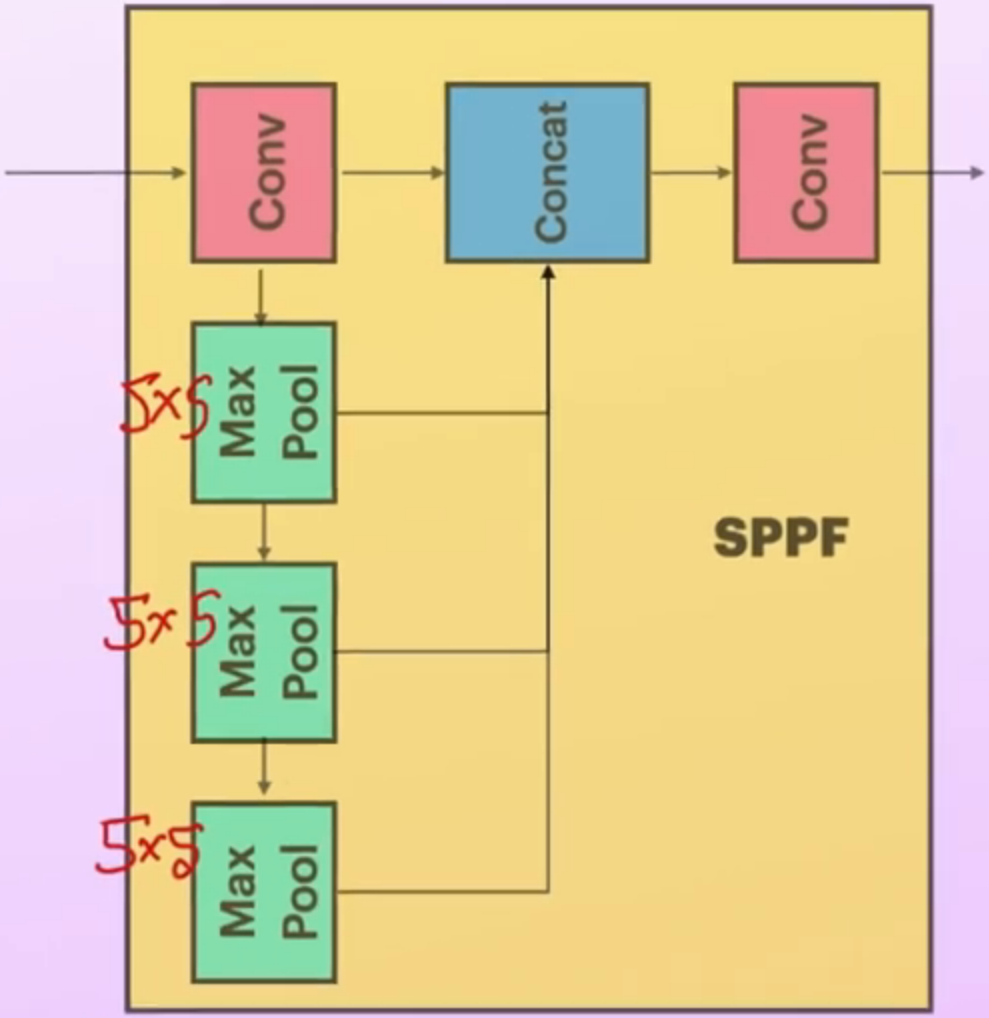

ected layers (fc6, fc7)
gth representation
spatial pyramid
pooling layer
feature maps of conv5
convolutional layers
put image


fc7)
n
spatial pyramid
pooling layer
feature maps of conv5
onal layers

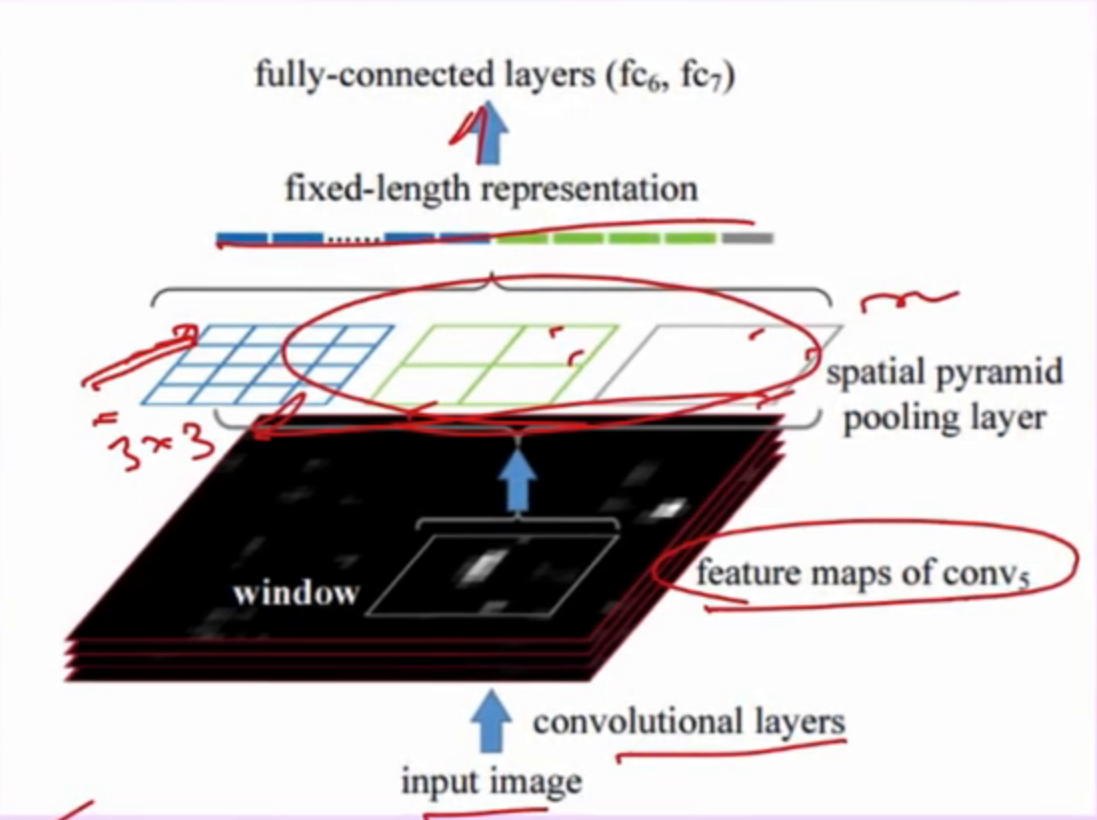

Conv
Concat
Conv
Max Pool
Max Pool
Max Pool
SPPF

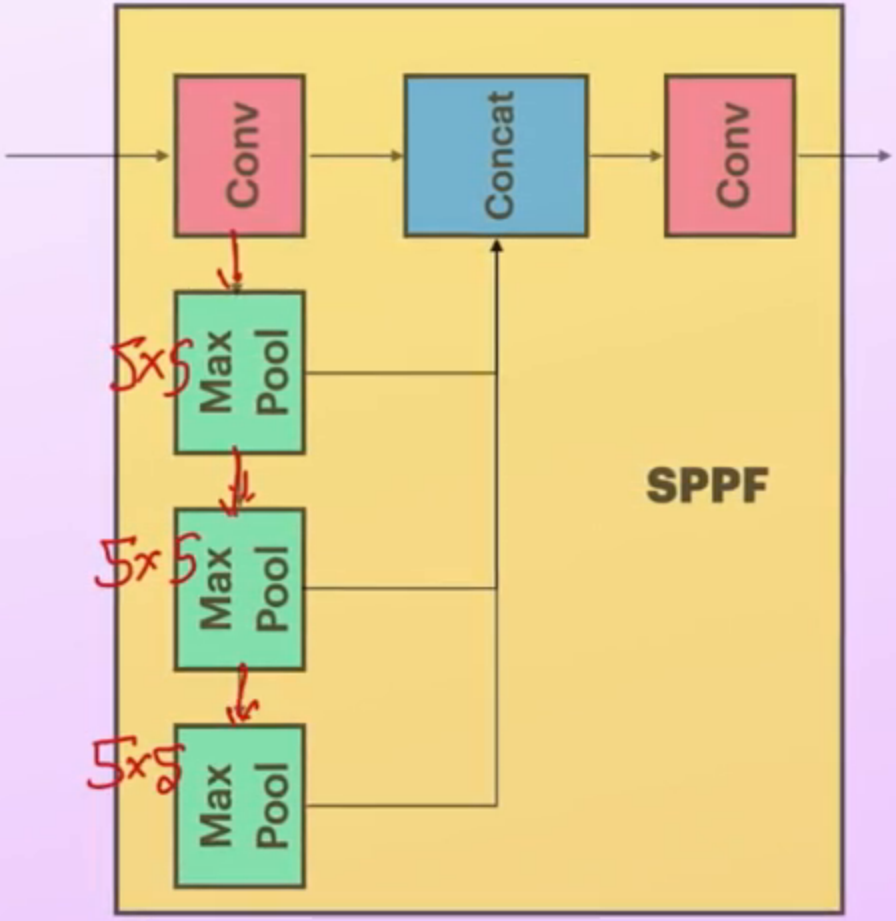
atial Pyramid Pooling Fast



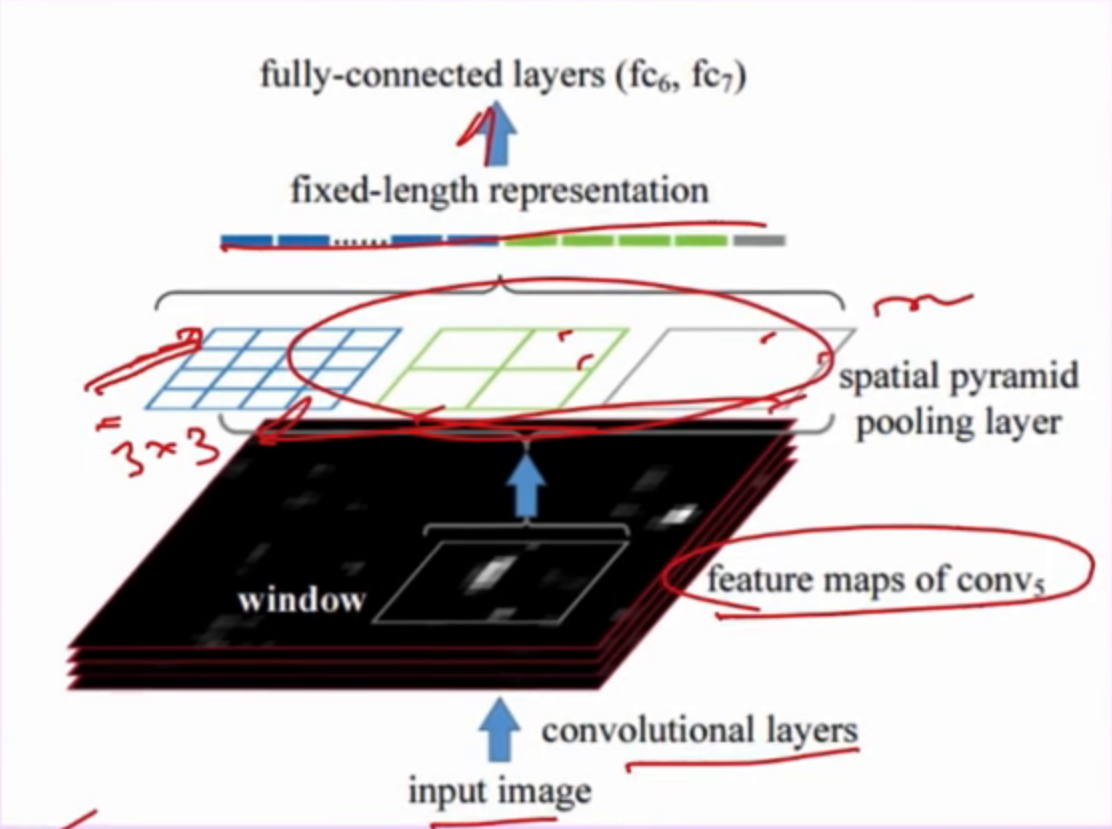


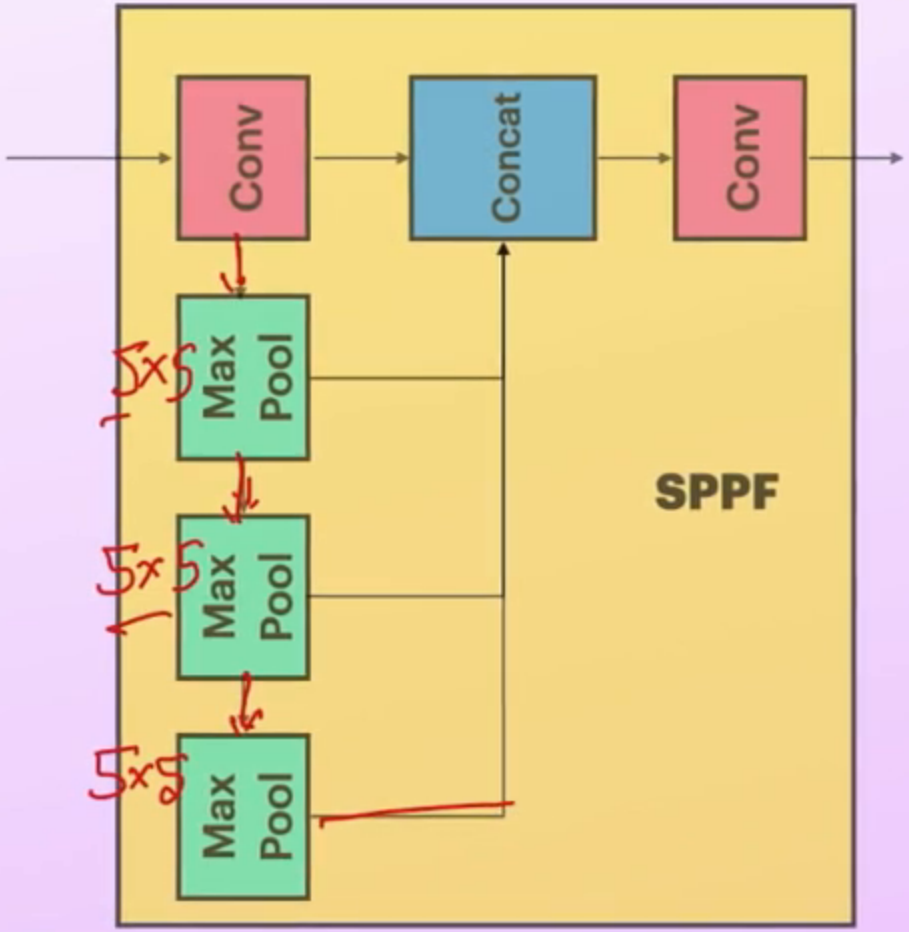
Spatial Pyramid Pooling Fast



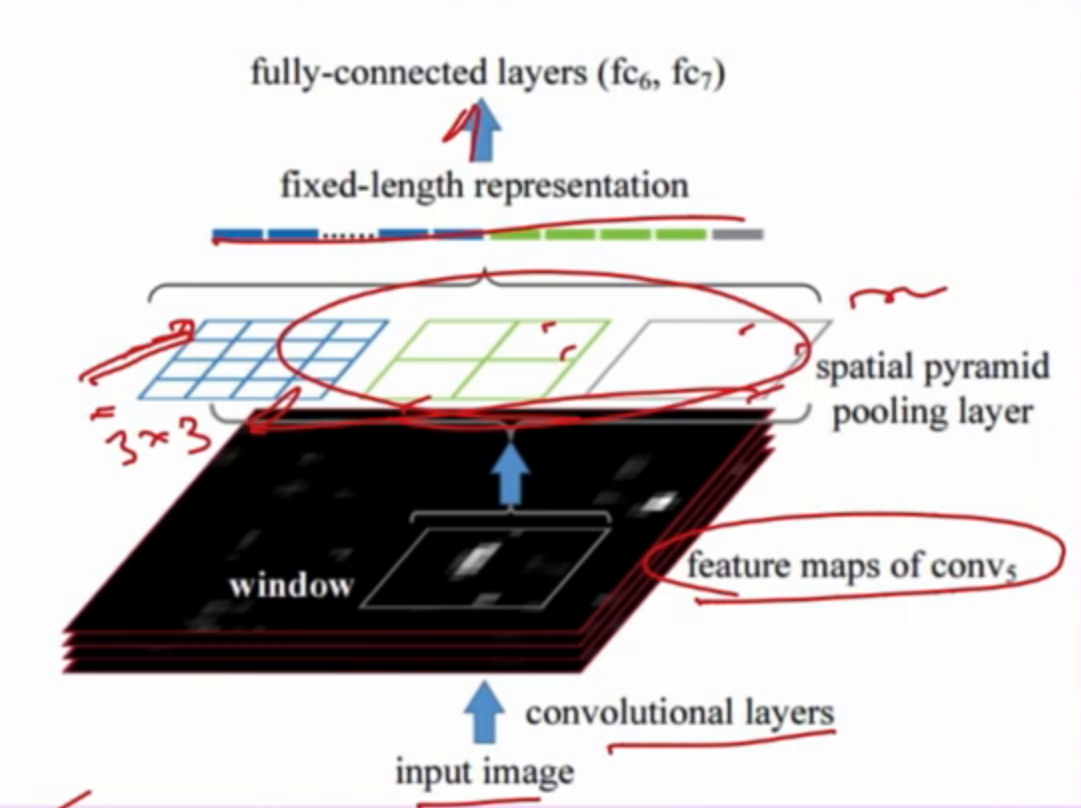


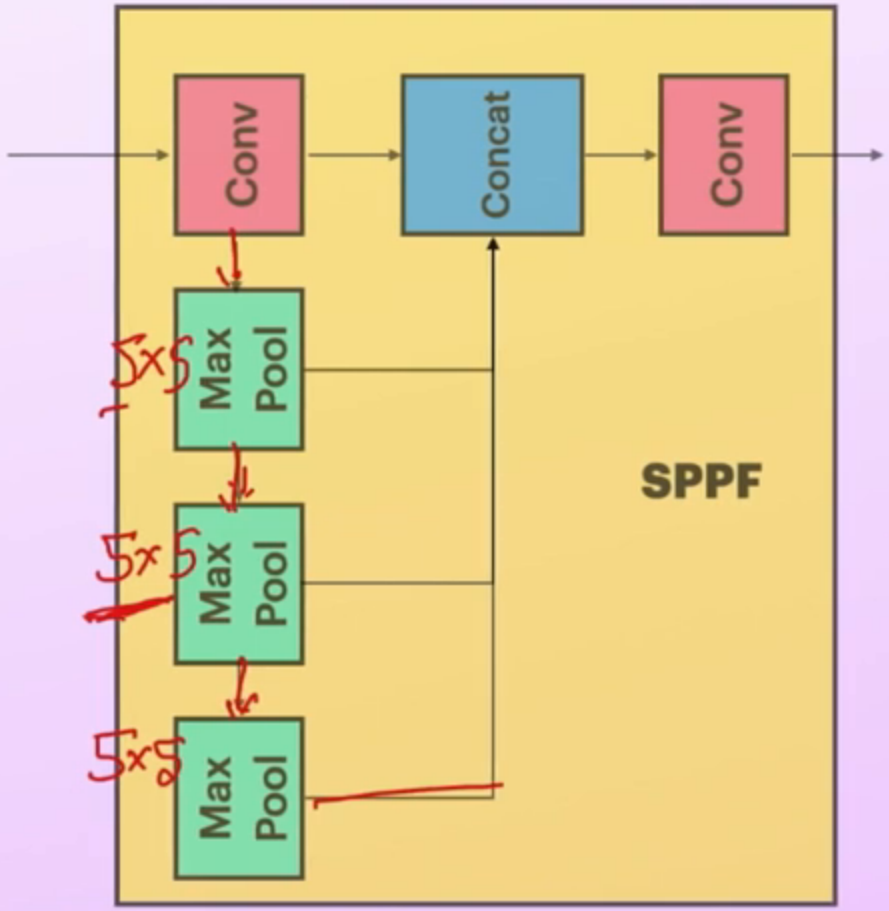
Spatial Pyramid Pooling Fast



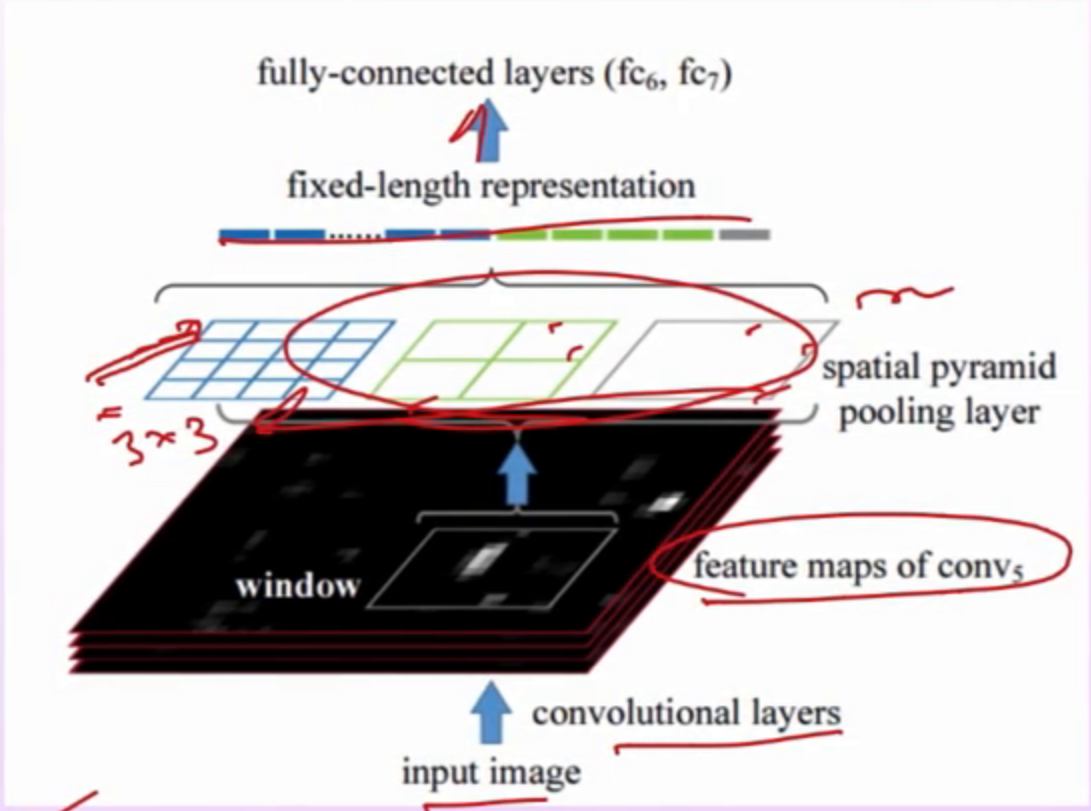


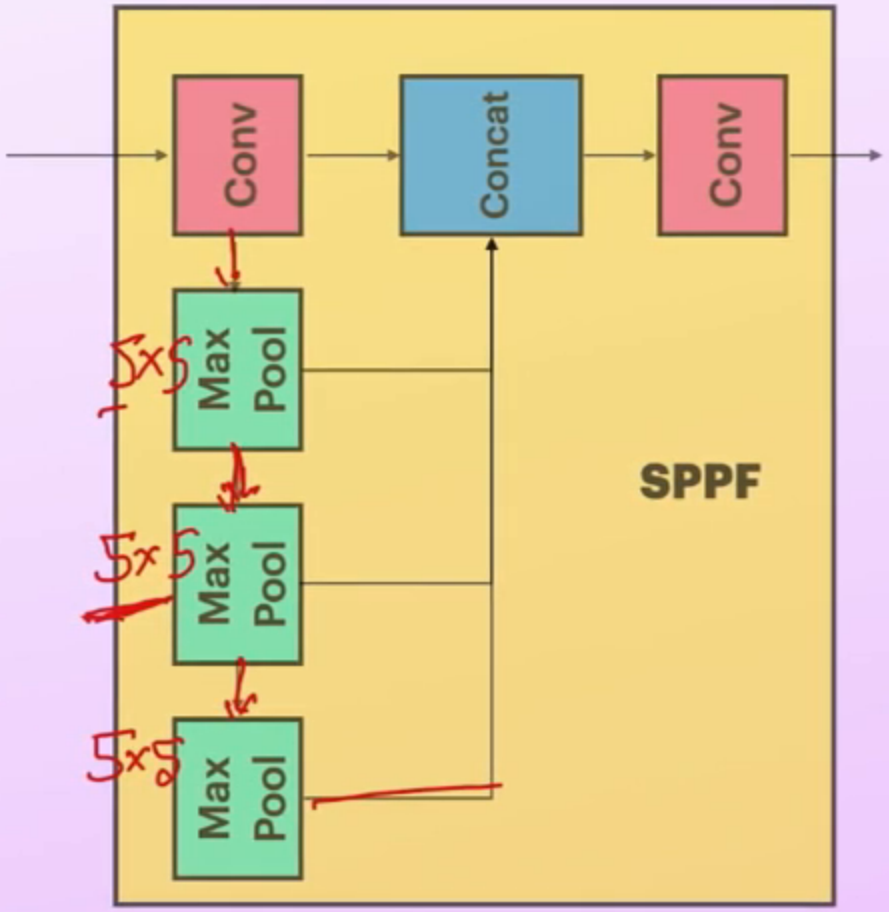
Spatial Pyramid Pooling Fast



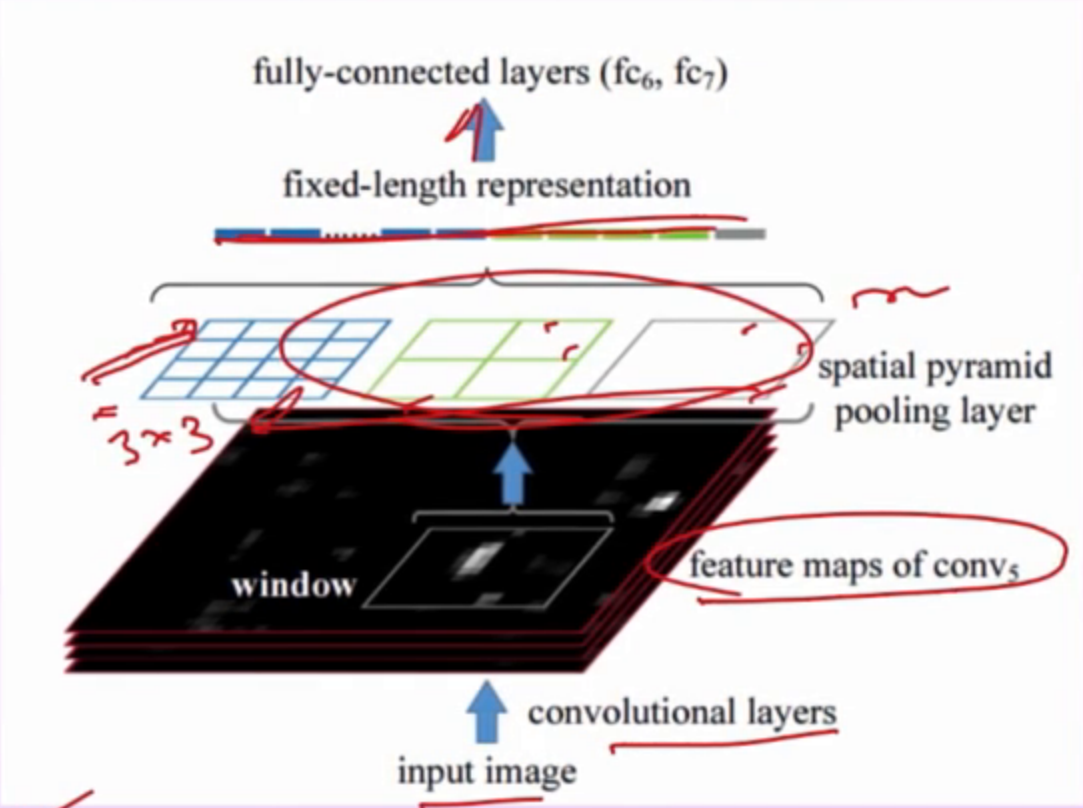


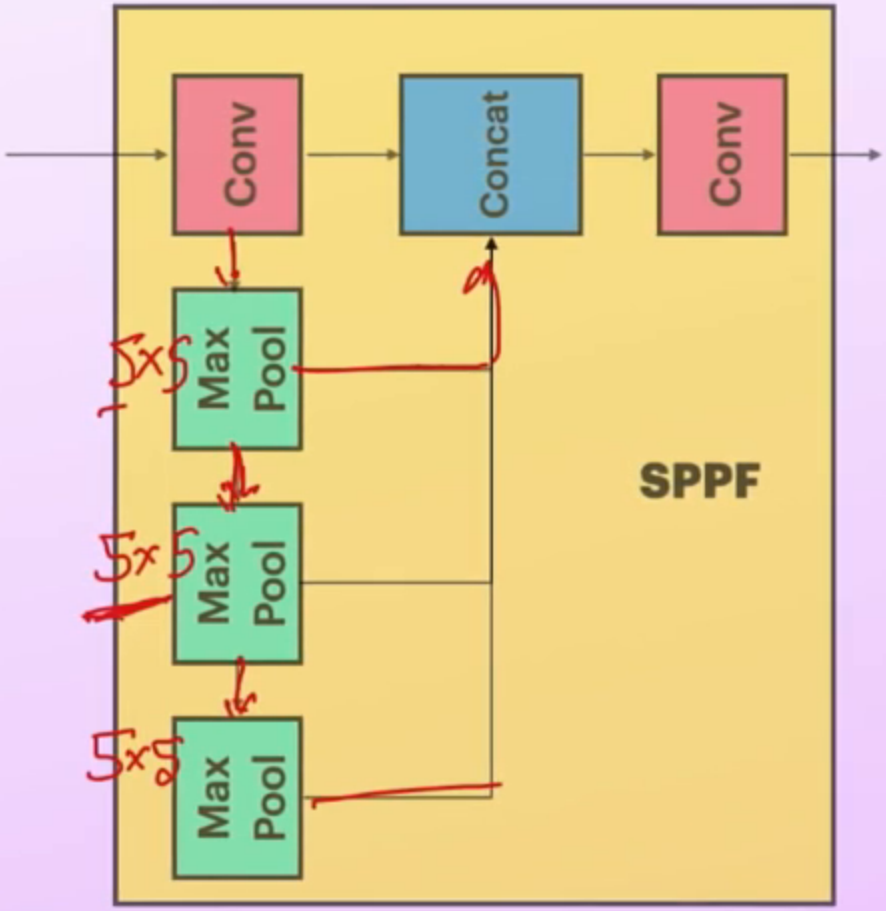
Spatial Pyramid Pooling Fast



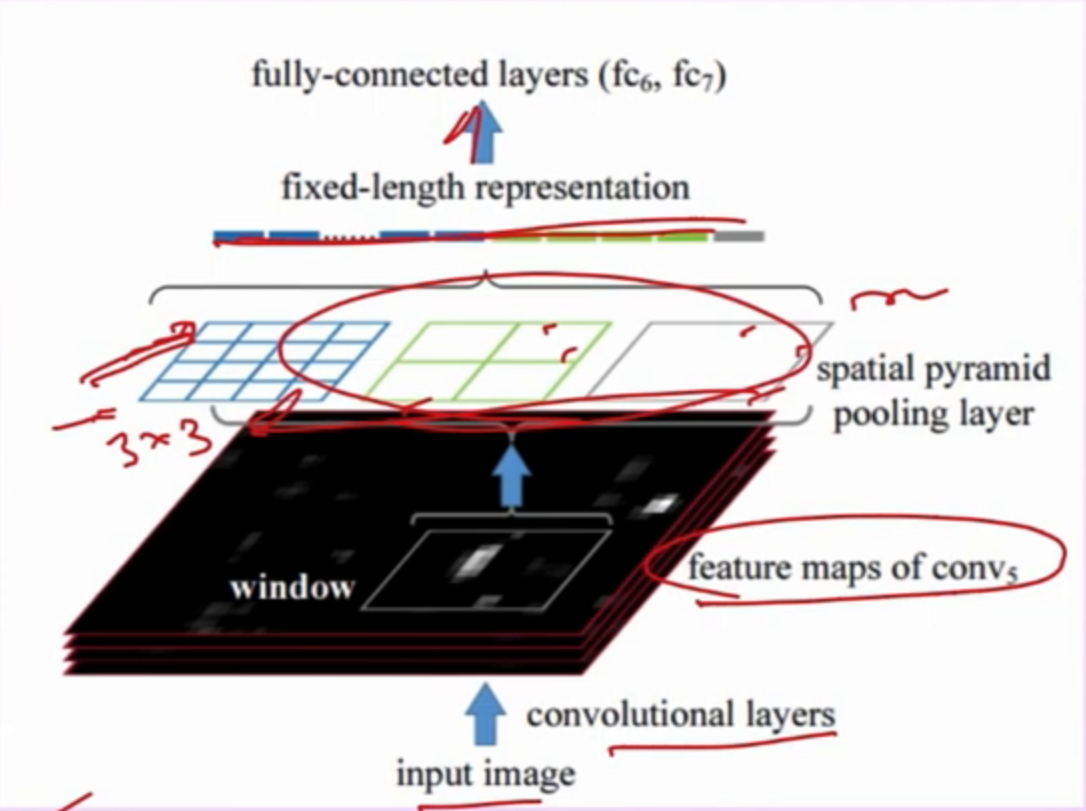


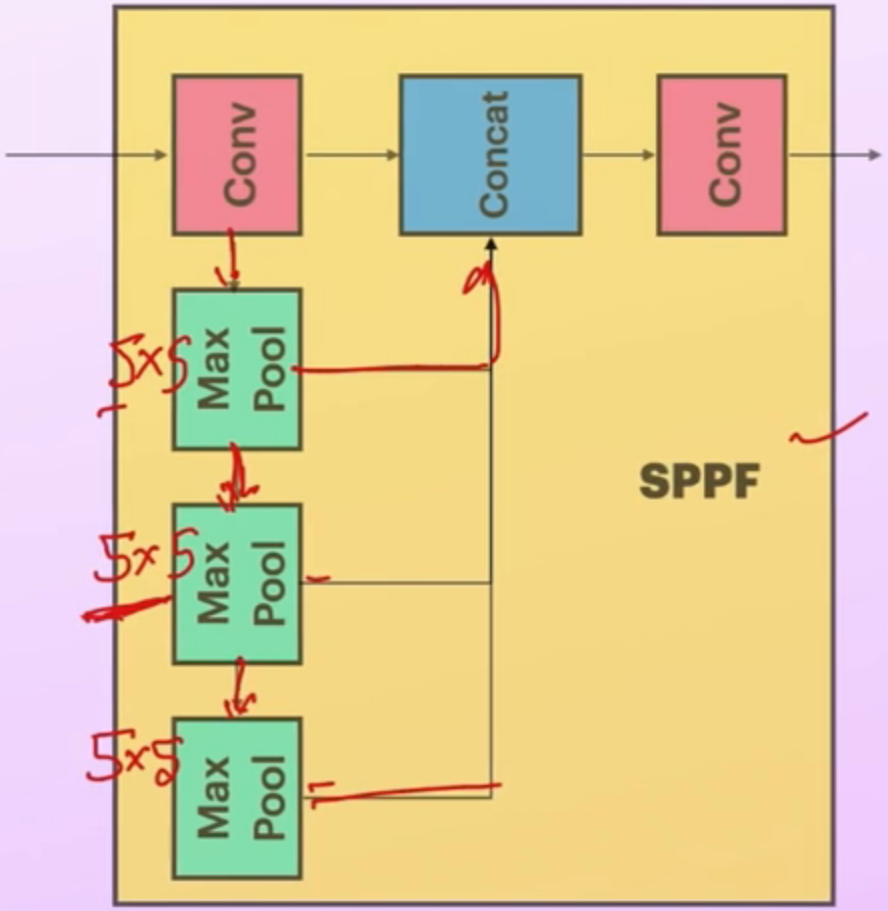
Spatial Pyramid Pooling Fast





Spatial Pyramid Pooling Fast





Path Aggregation Network

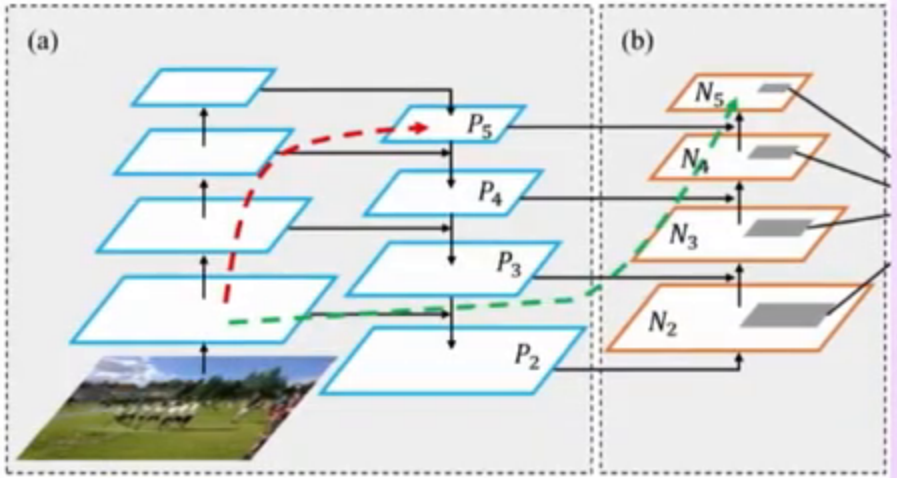


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature

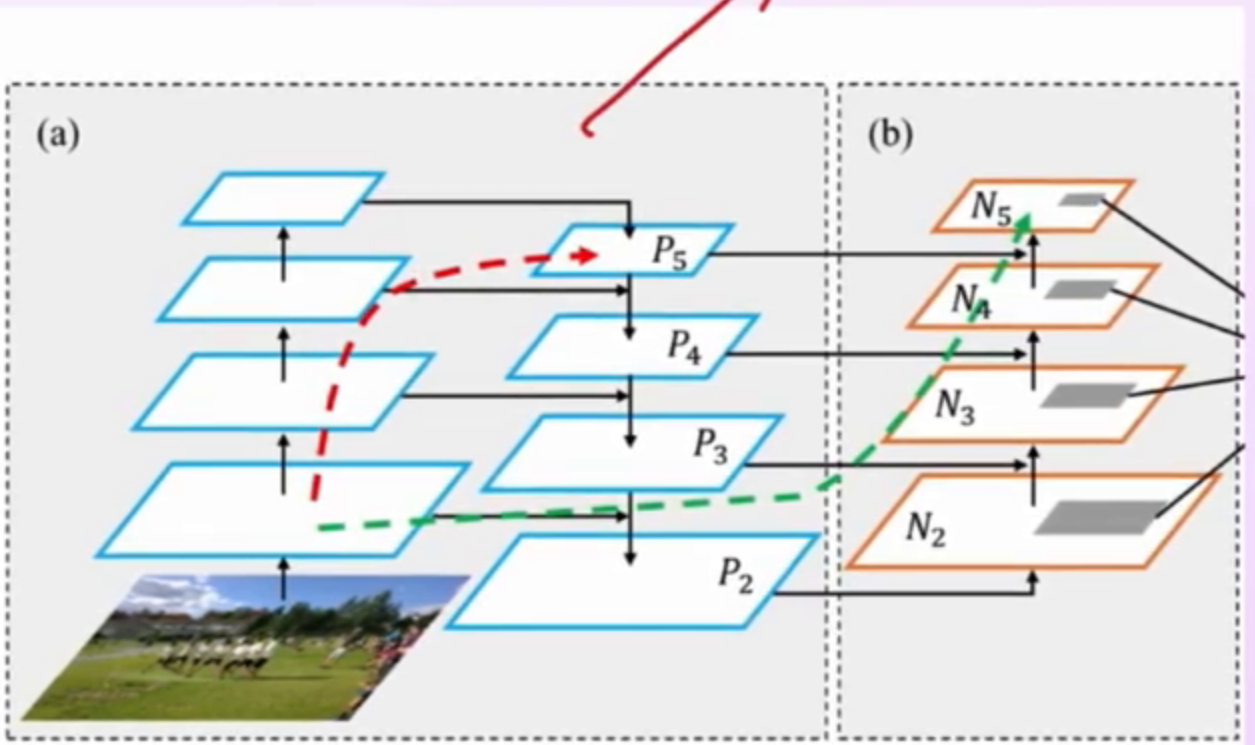


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature

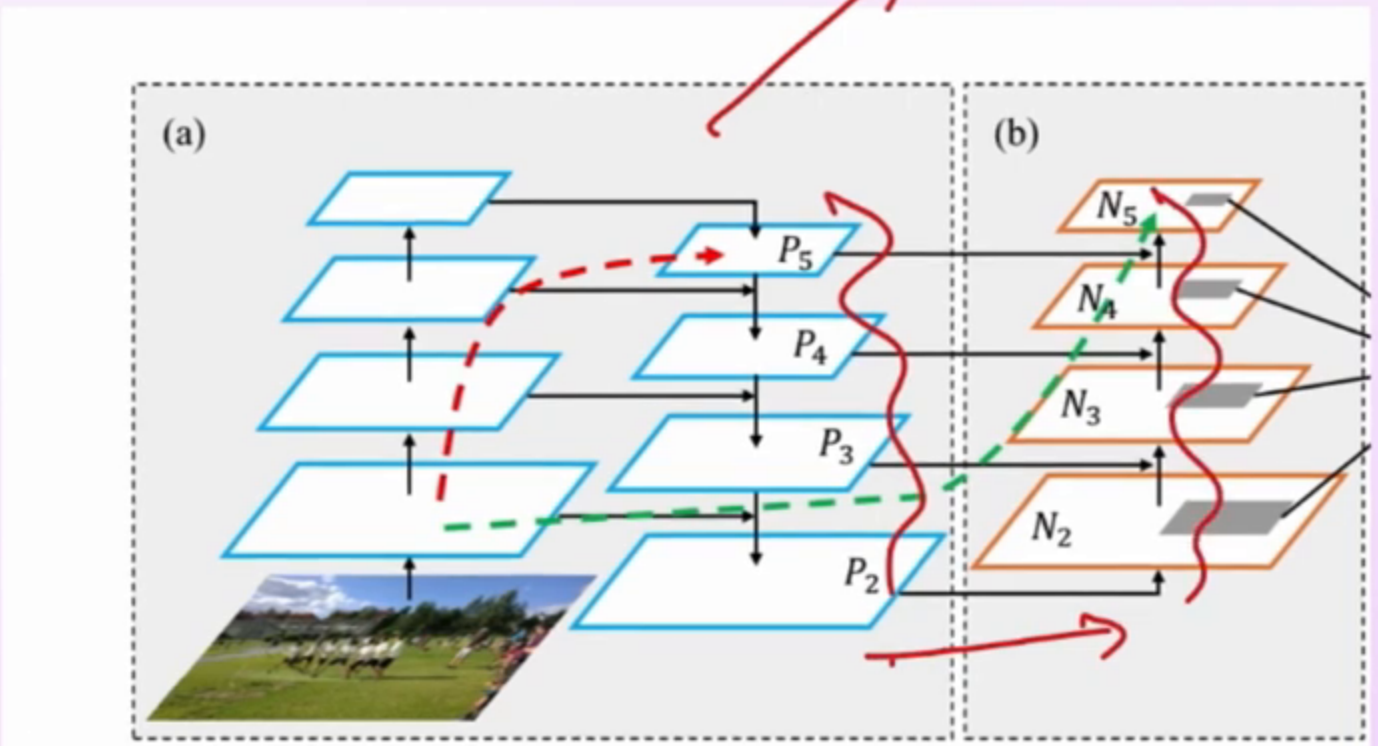


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature

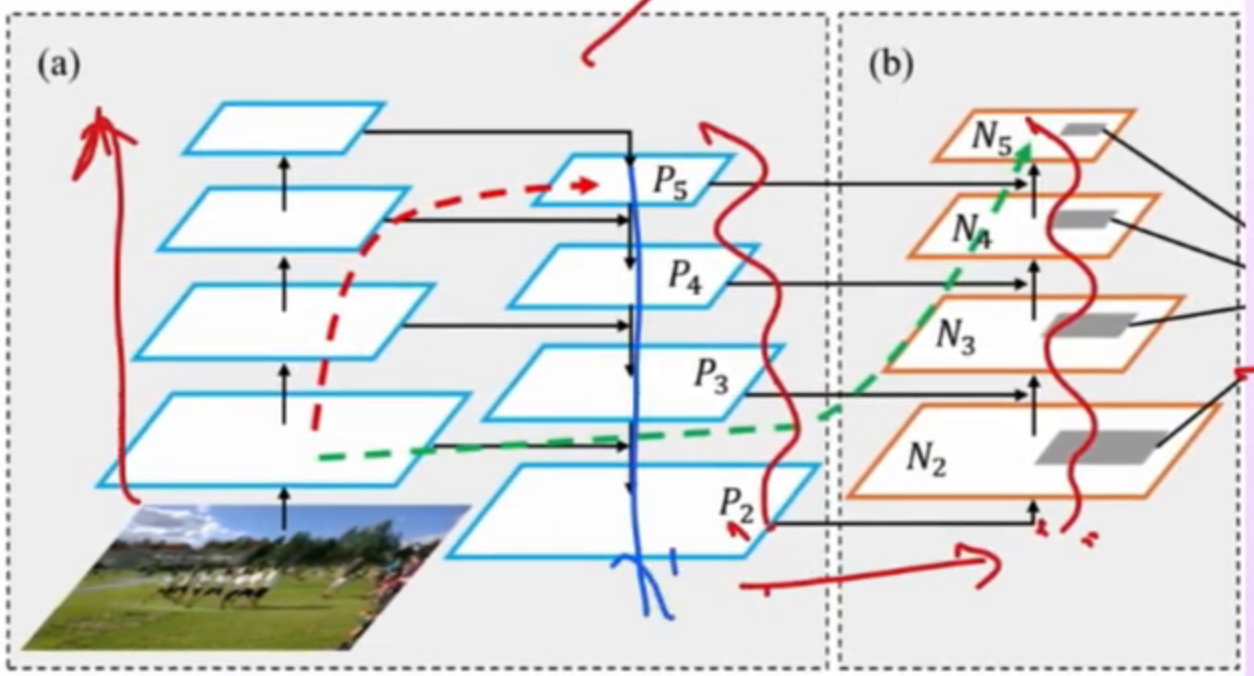


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature

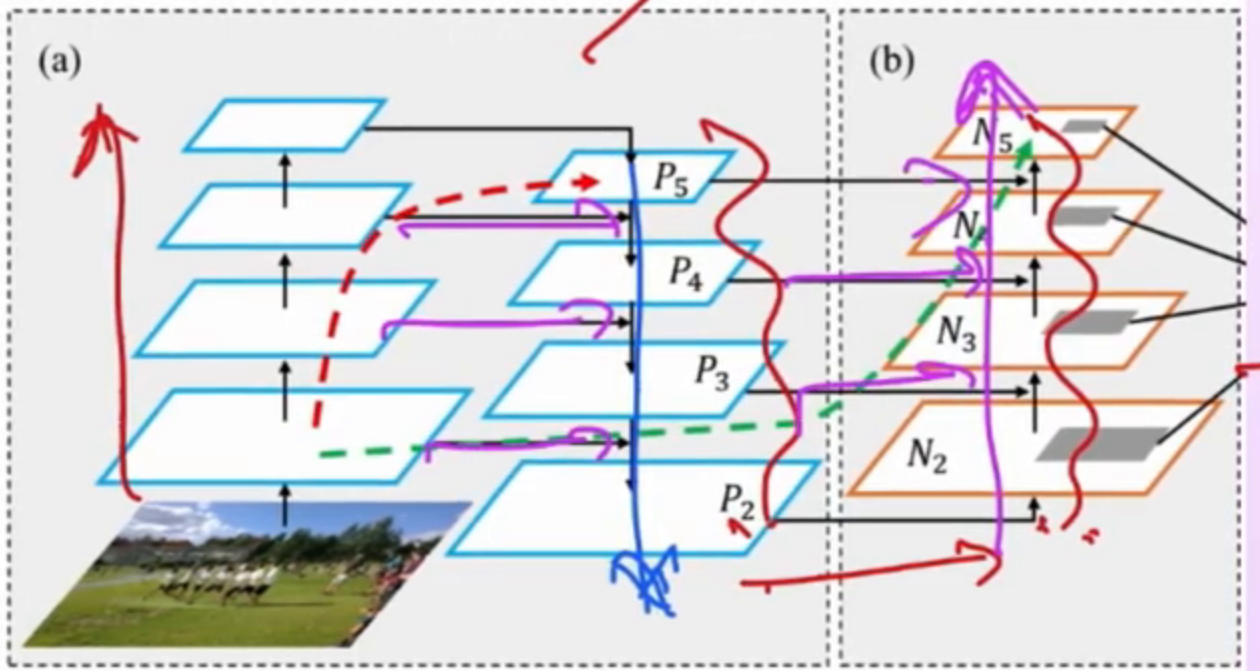


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature

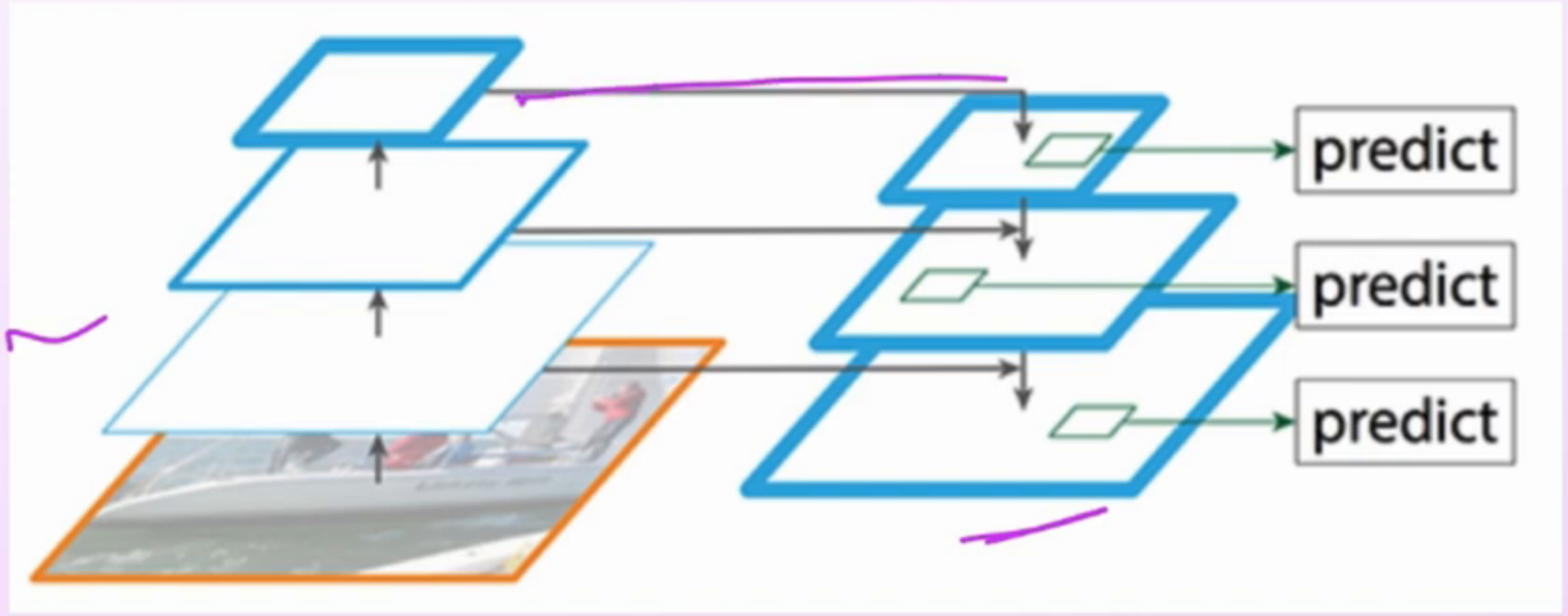
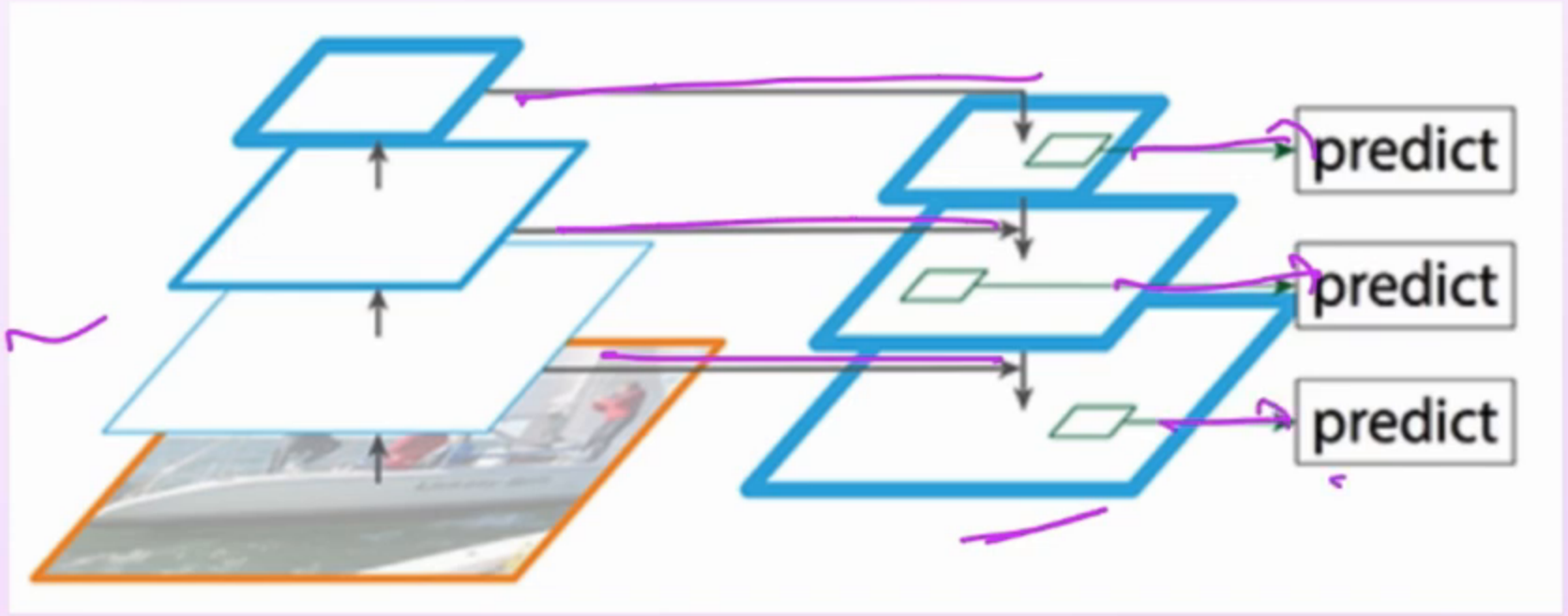


Figure 1. Illustration of our framev
branch. (e) Fully-connected fusion.



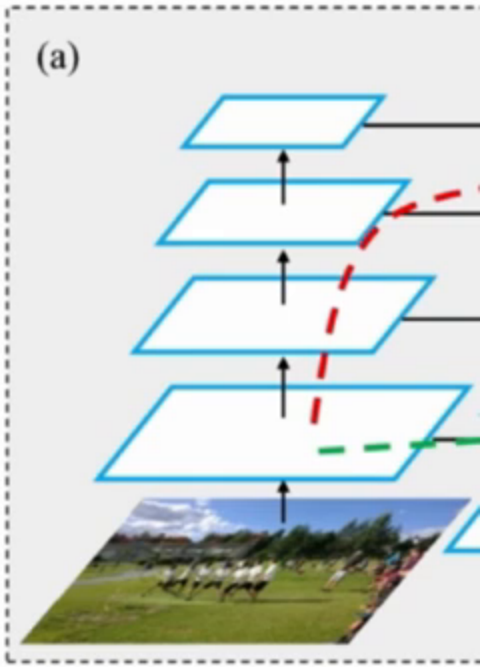


Figure 1. Illustration of our framev
branch. (e) Fully-connected fusion.

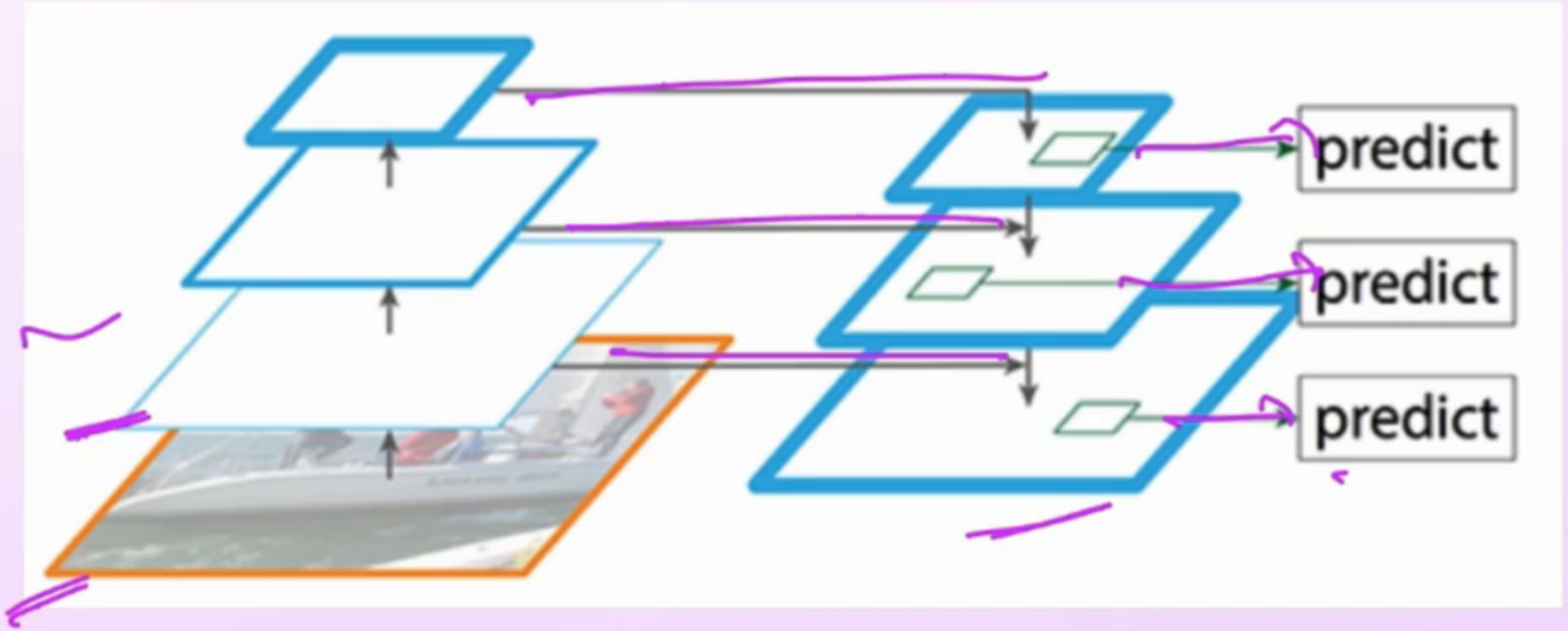
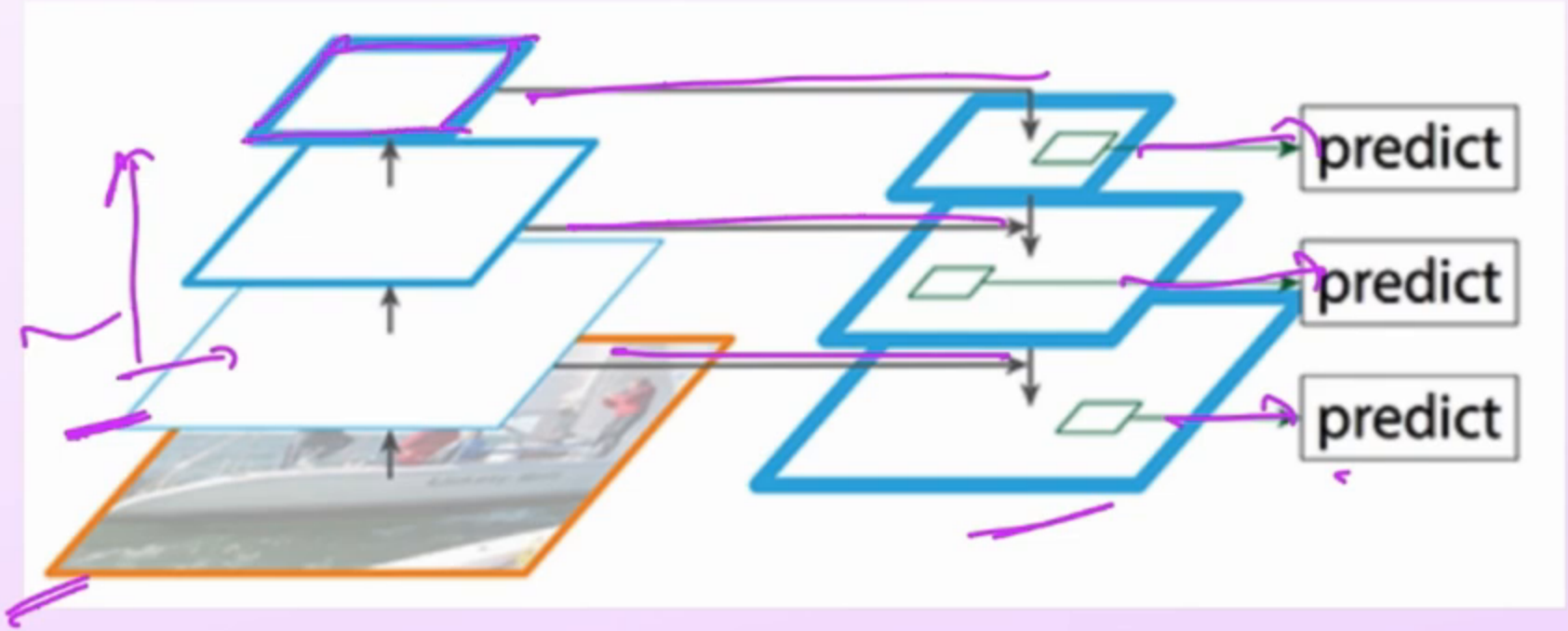


Figure 1. Illustration of our framev
branch. (e) Fully-connected fusion.



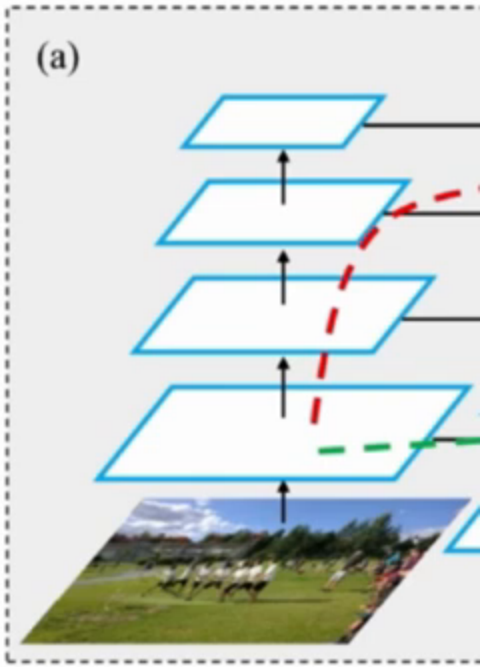


Figure 1. Illustration of our framev
branch. (e) Fully-connected fusion.

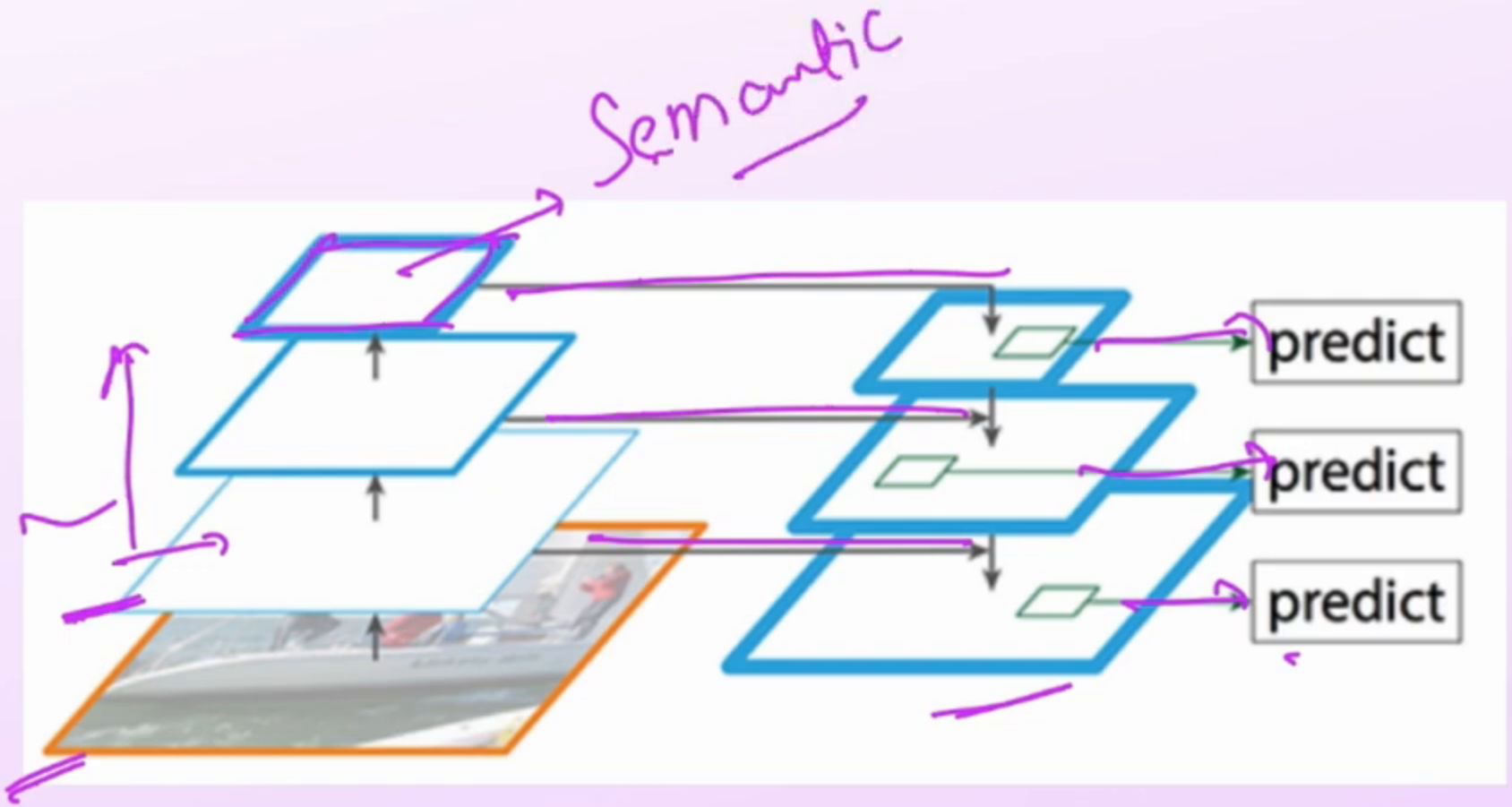


Figure 1. Illustration of our framev
branch. (e) Fully-connected fusion.

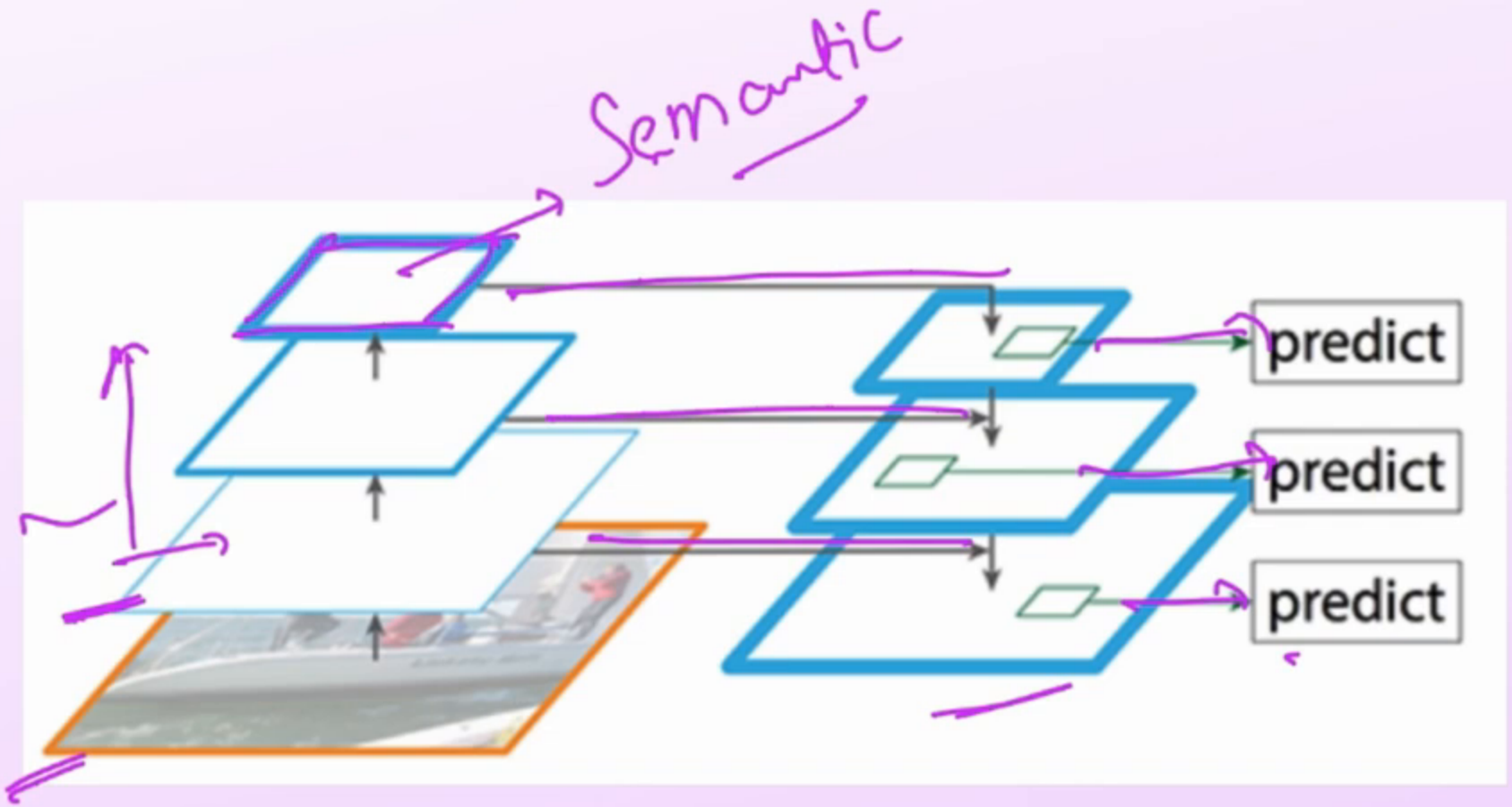


Figure 1. Illustration of our framev
branch. (e) Fully-connected fusion.

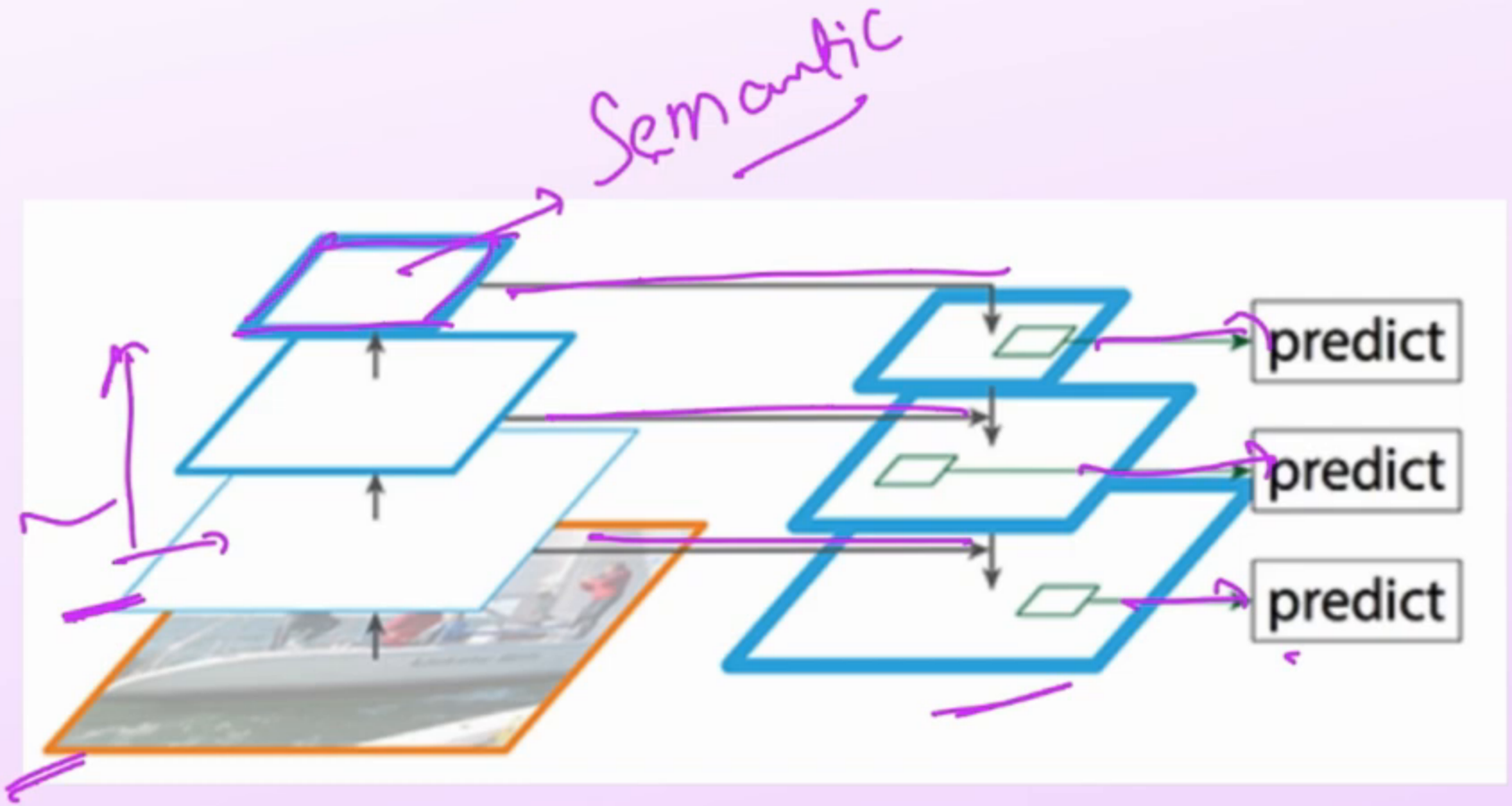


Figure 1. Illustration of our framev
branch. (e) Fully-connected fusion.

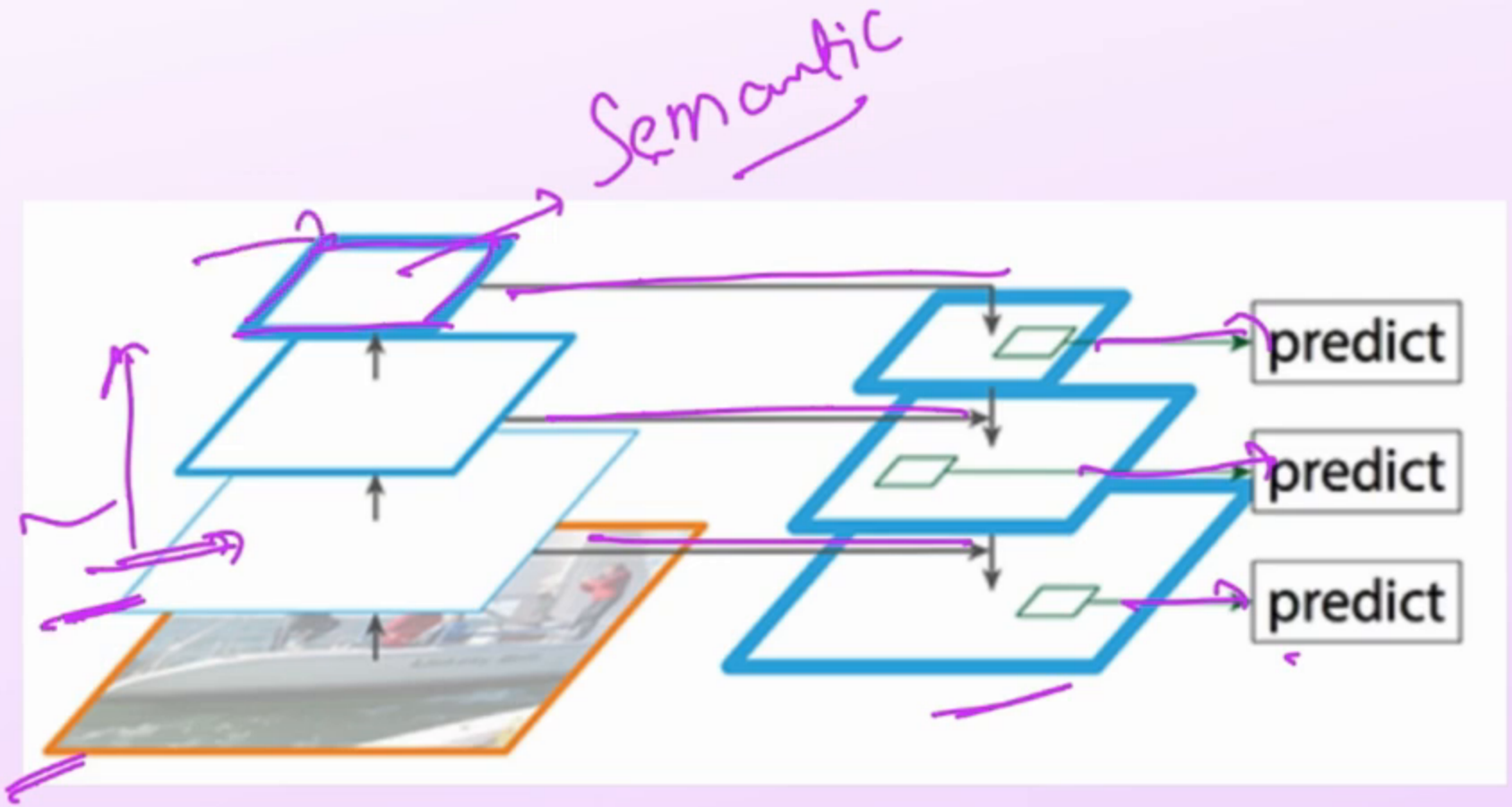


Figure 1. Illustration of our framev
branch. (e) Fully-connected fusion.

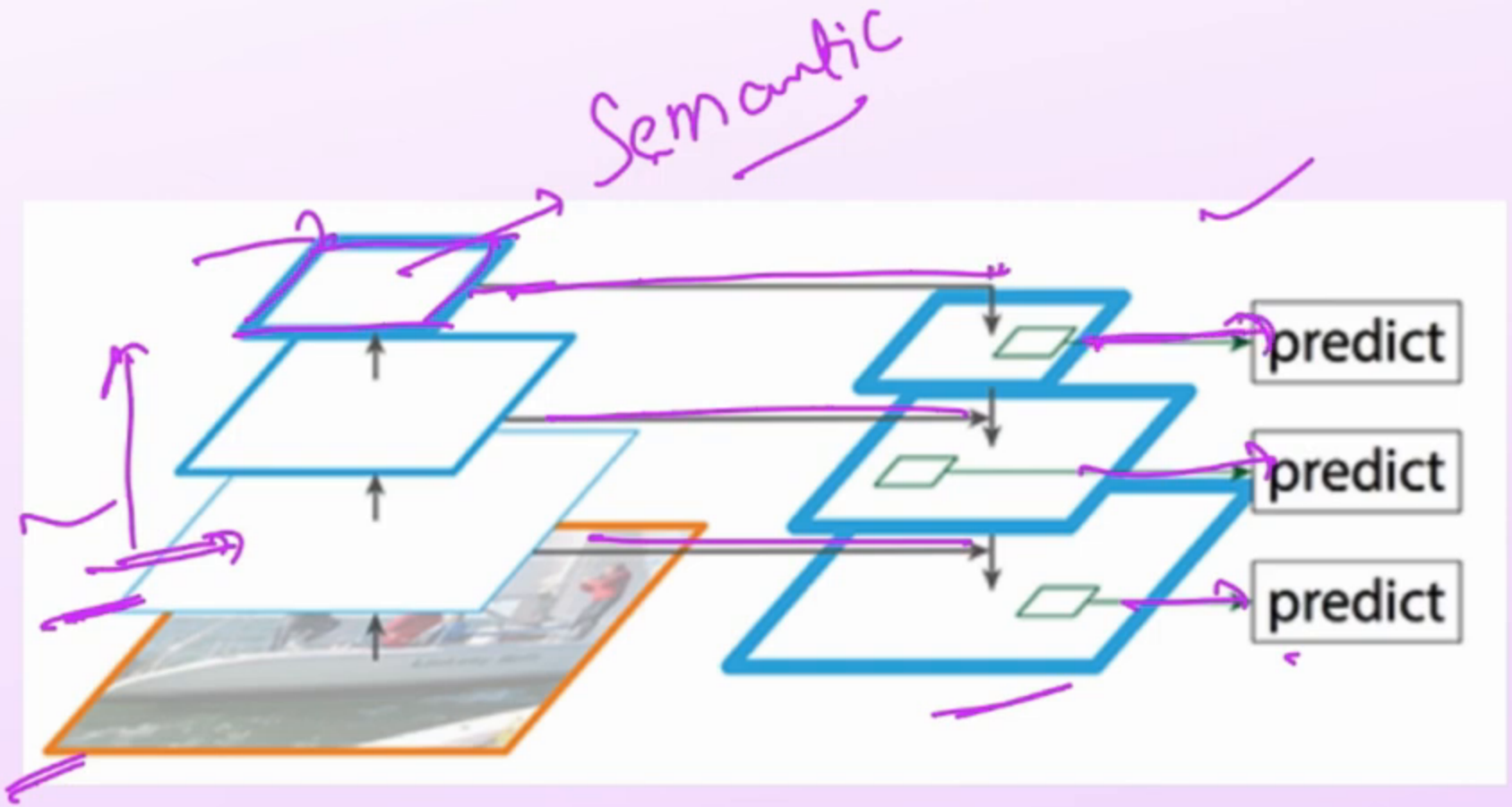
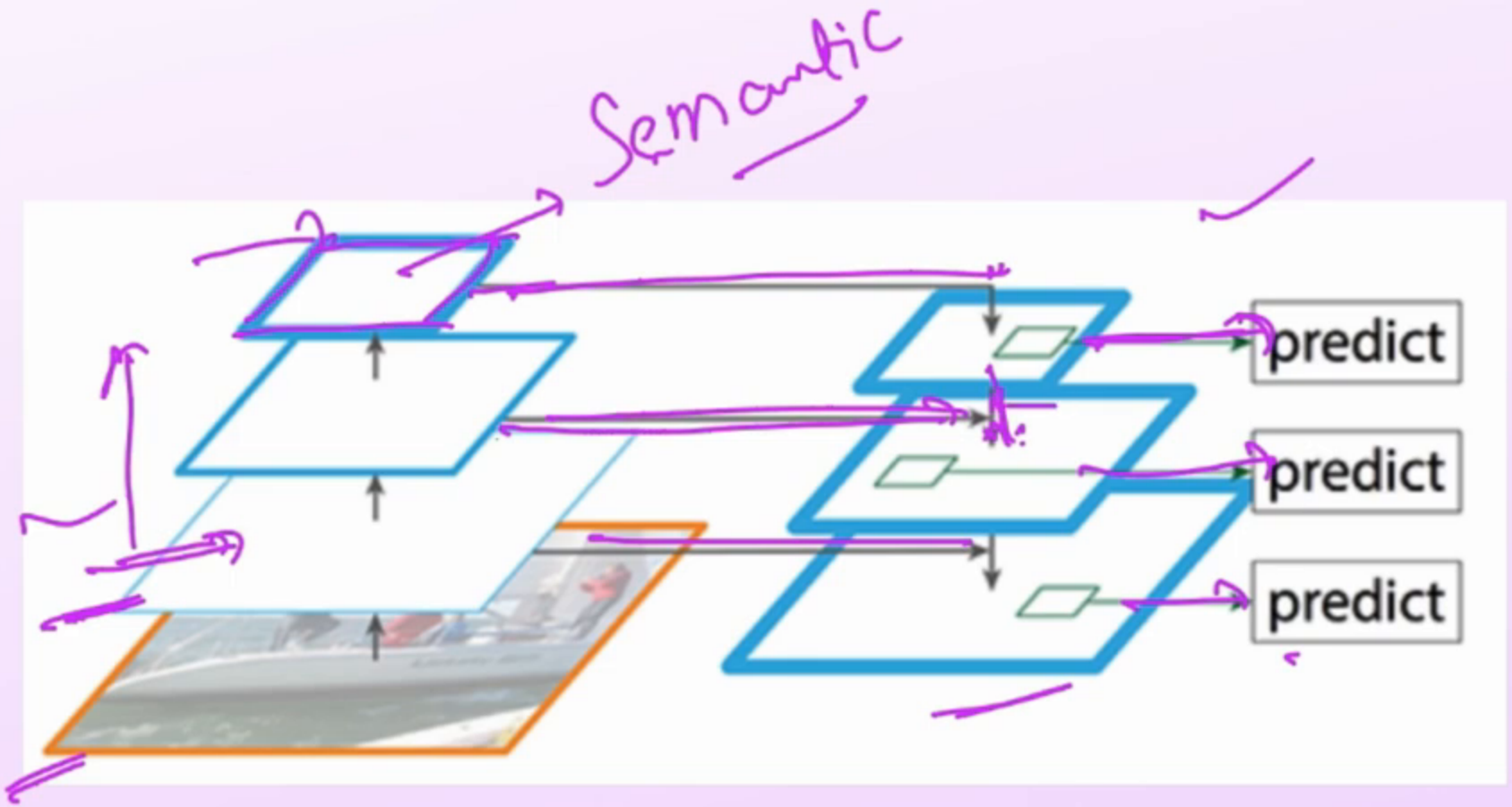


Figure 1. Illustration of our framev
branch. (e) Fully-connected fusion.



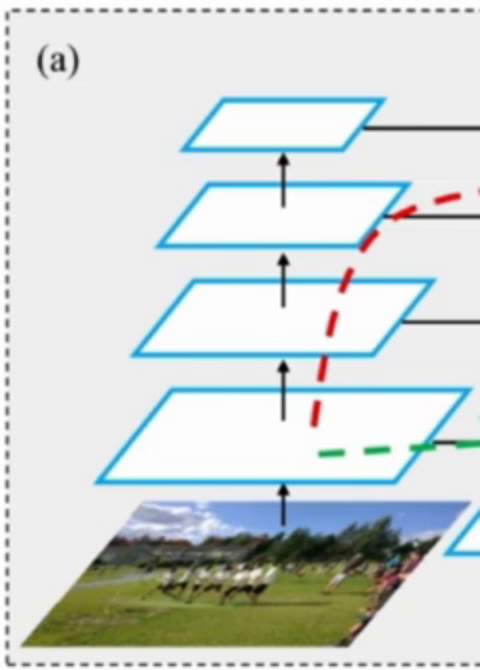


Figure 1. Illustration of our framev
branch. (e) Fully-connected fusion.

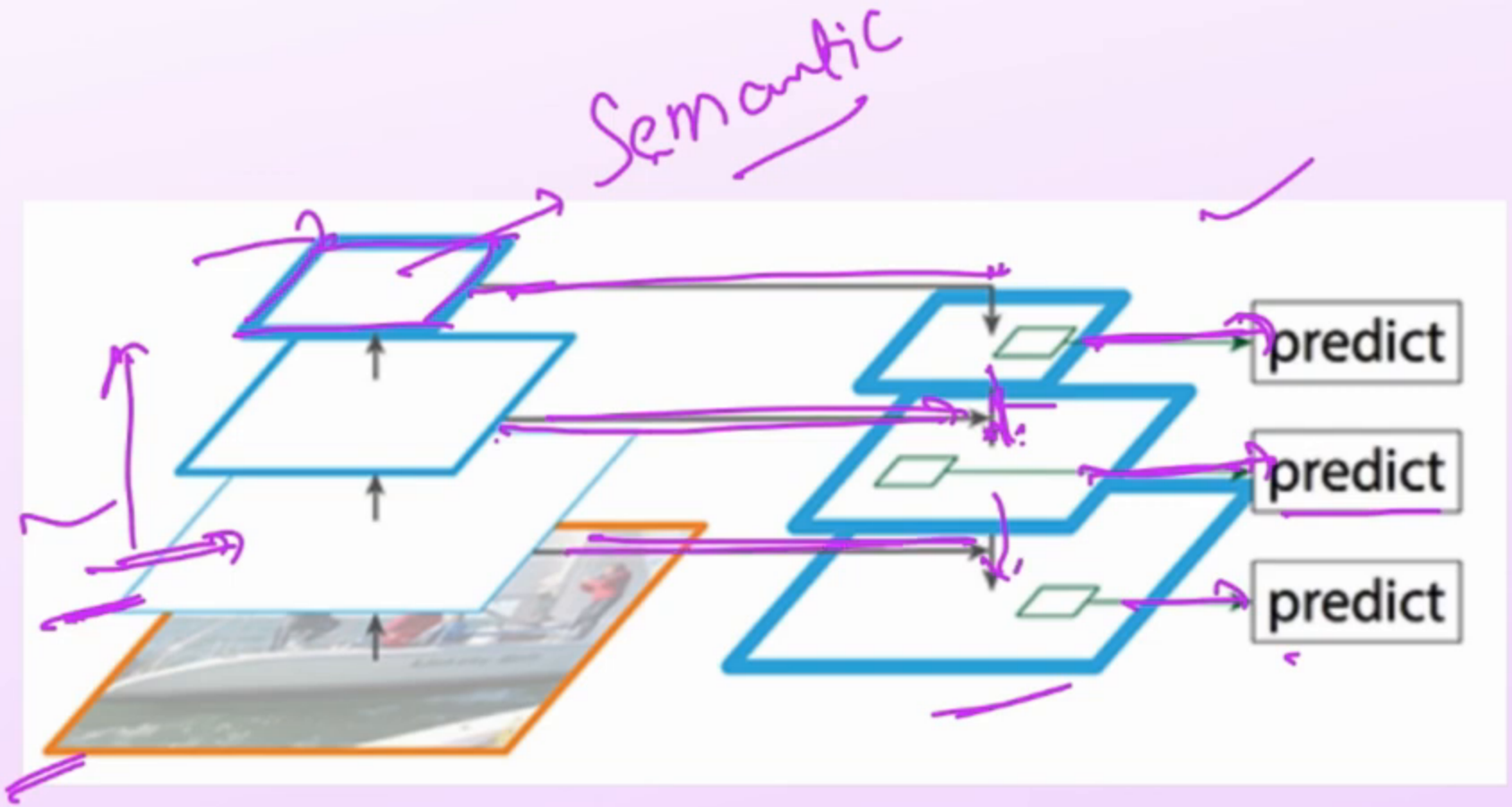


Figure 1. Illustration of our framev
branch. (e) Fully-connected fusion.

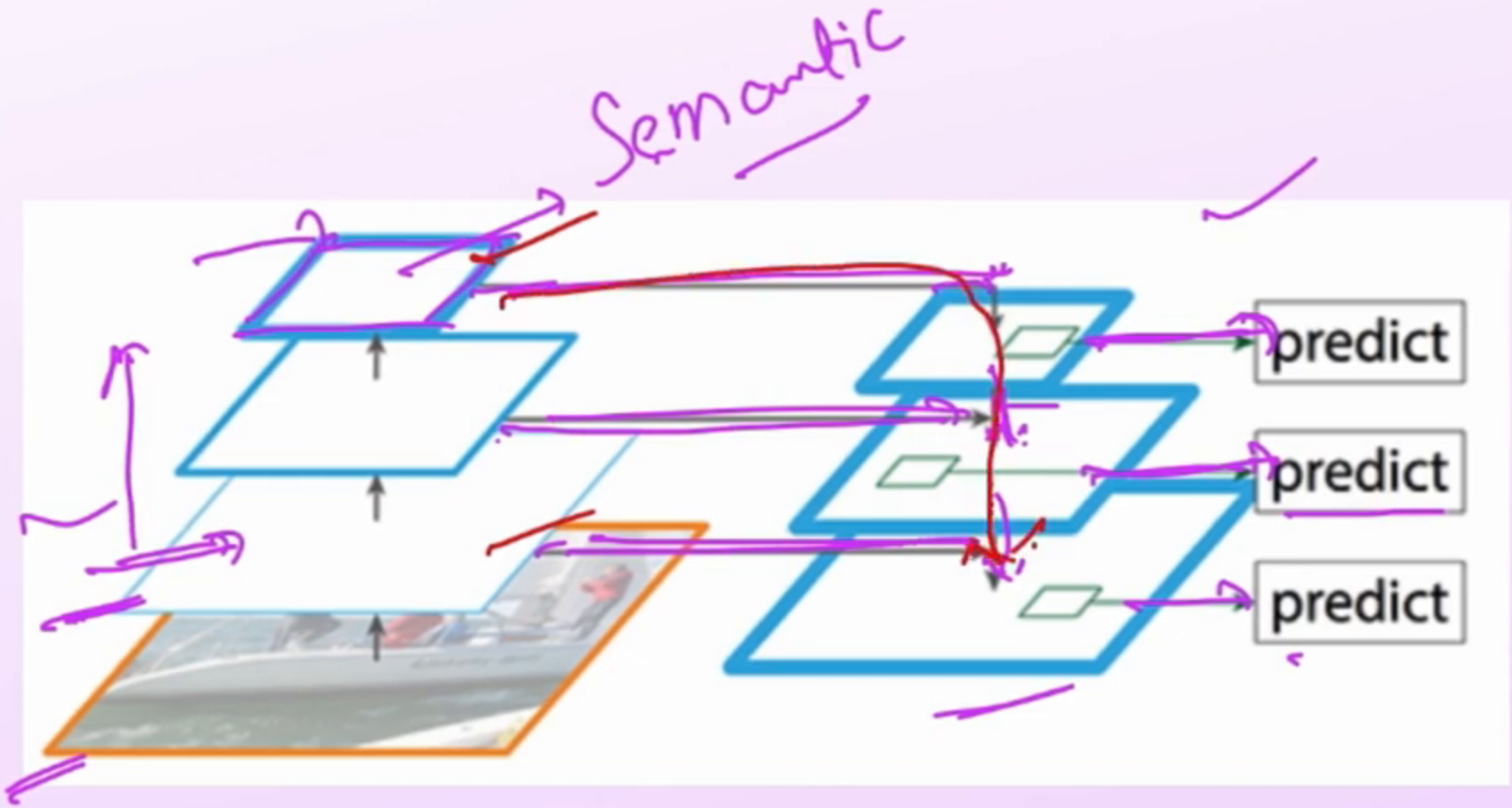
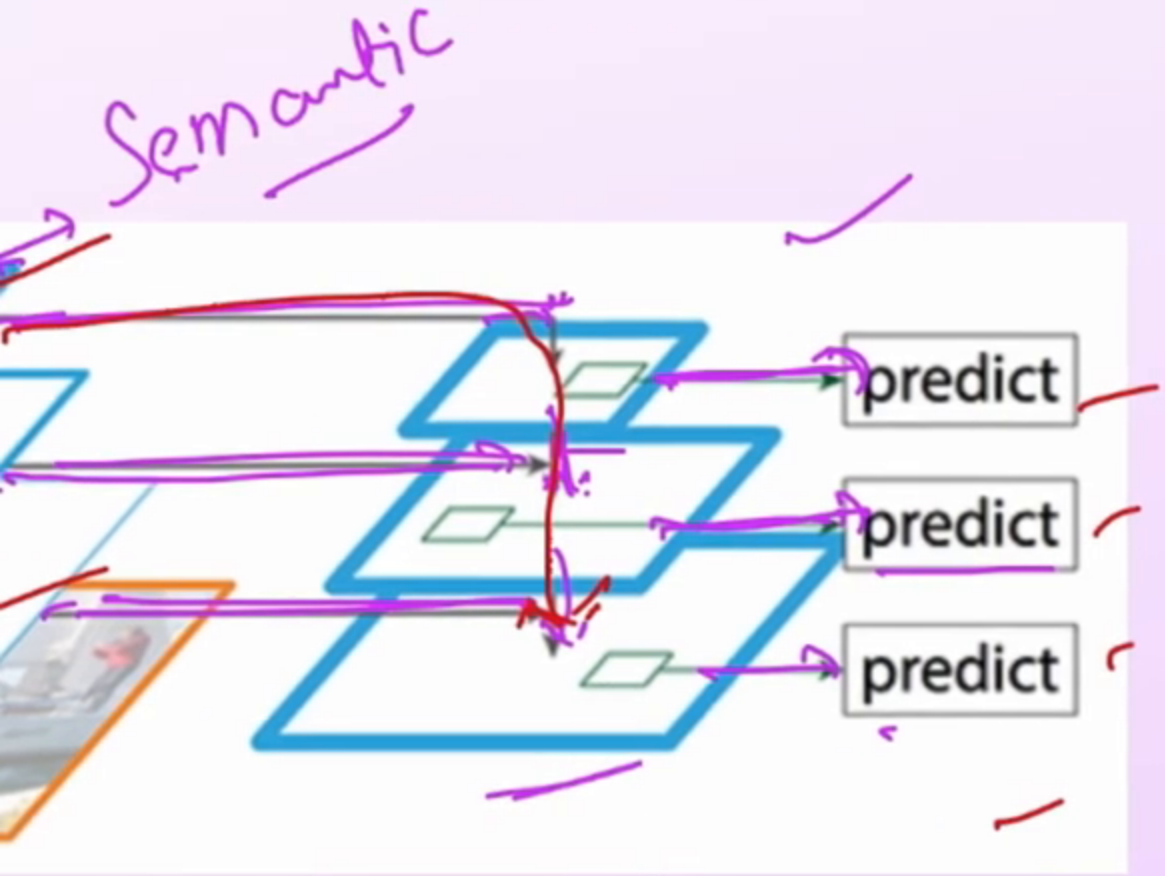


Figure 1. Illustration of our framev
branch. (e) Fully-connected fusion.



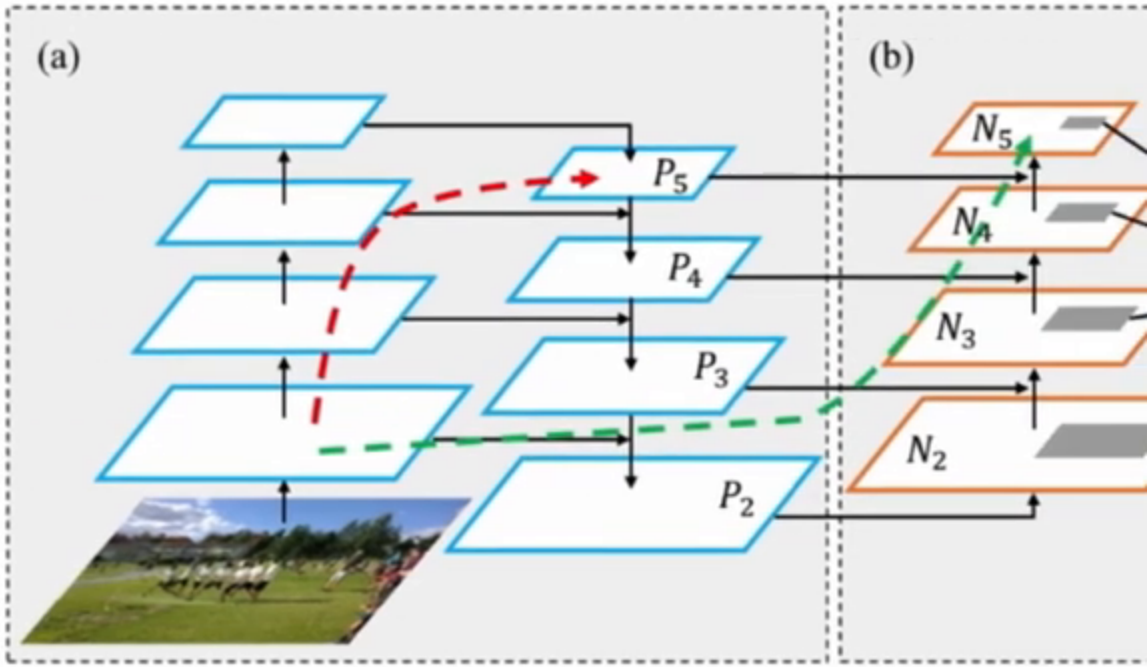
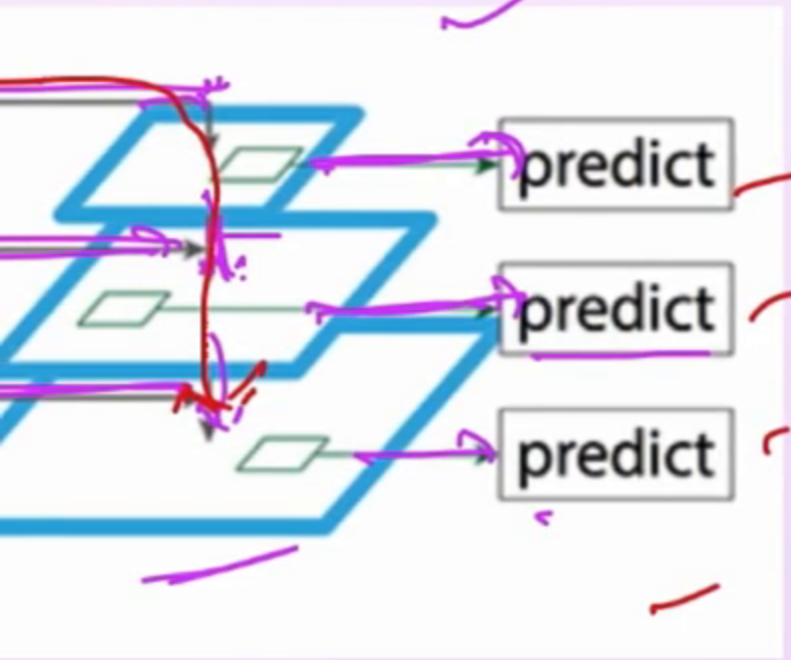


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up
branch. (e) Fully-connected fusion. Note that we omit channel dimension of



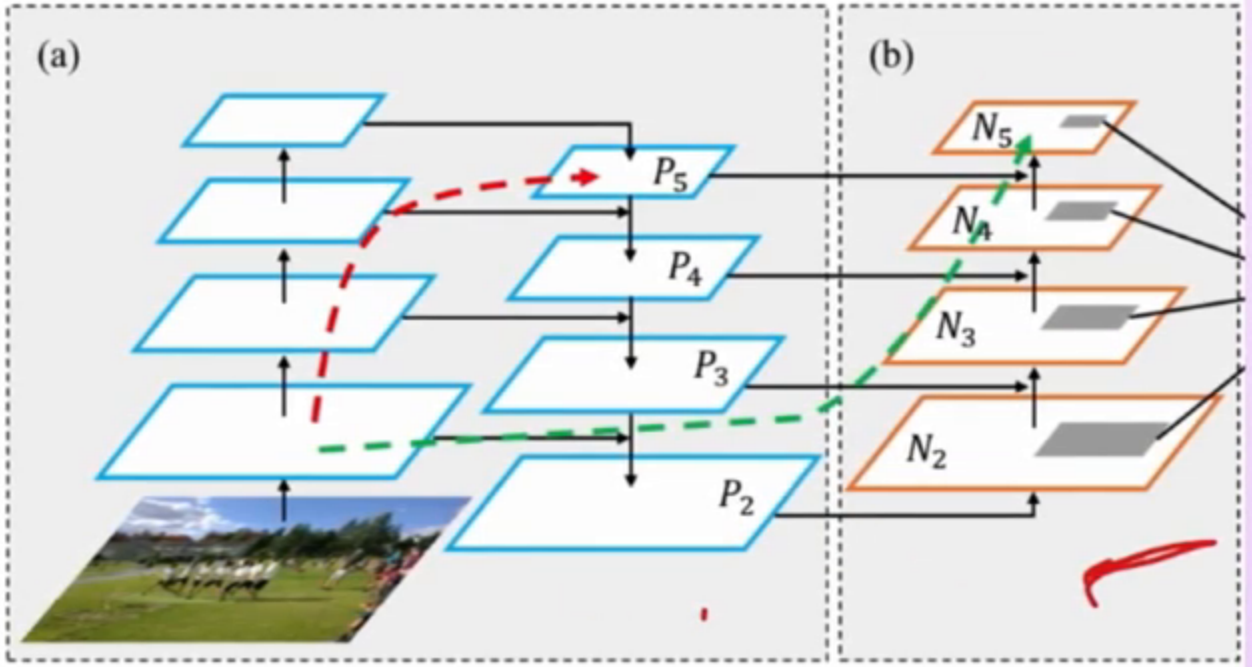
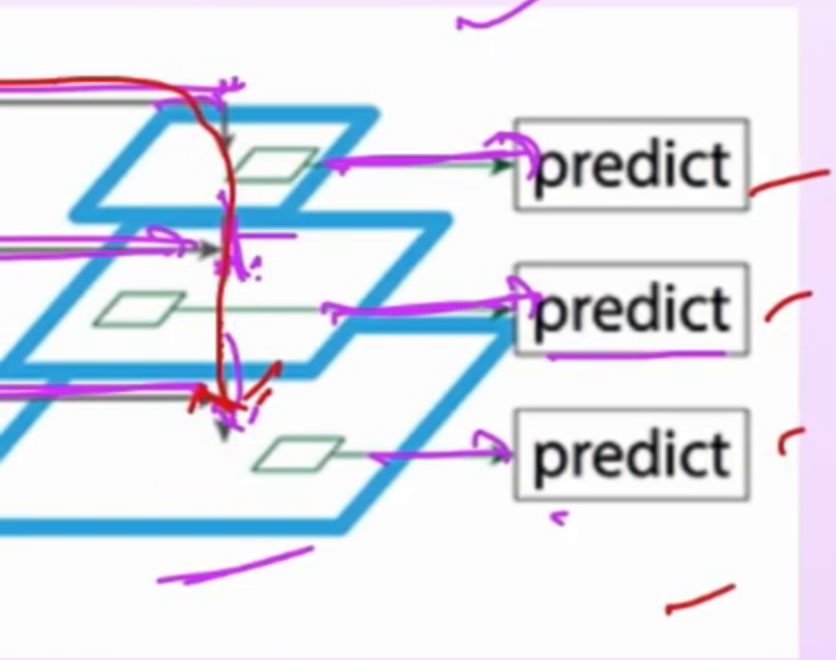


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



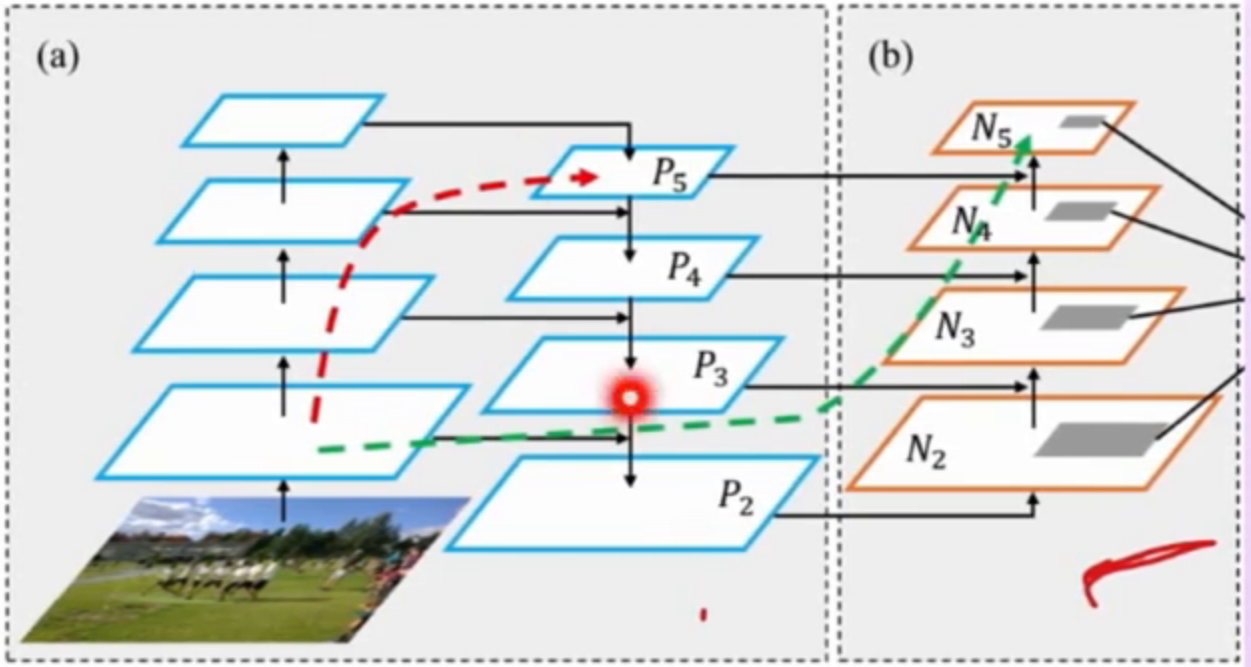
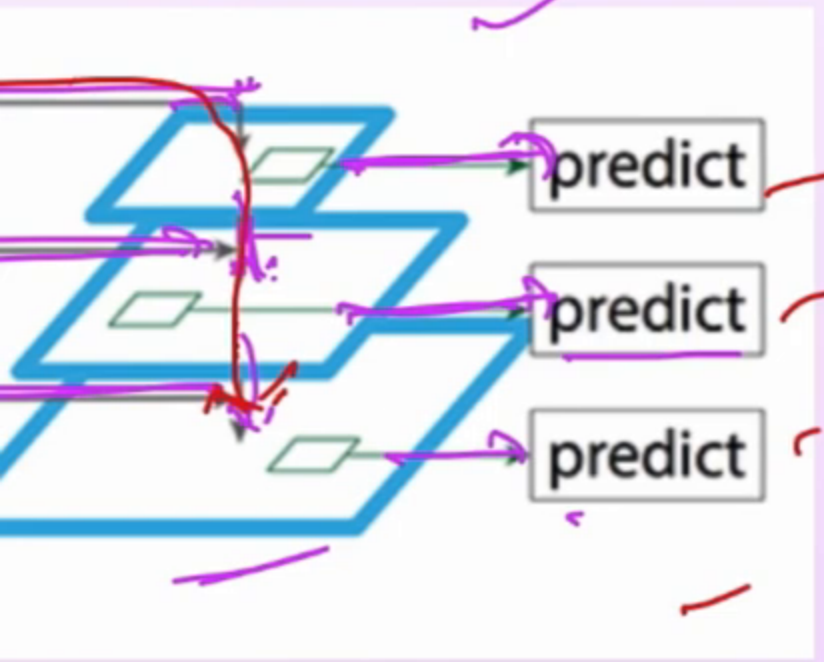


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



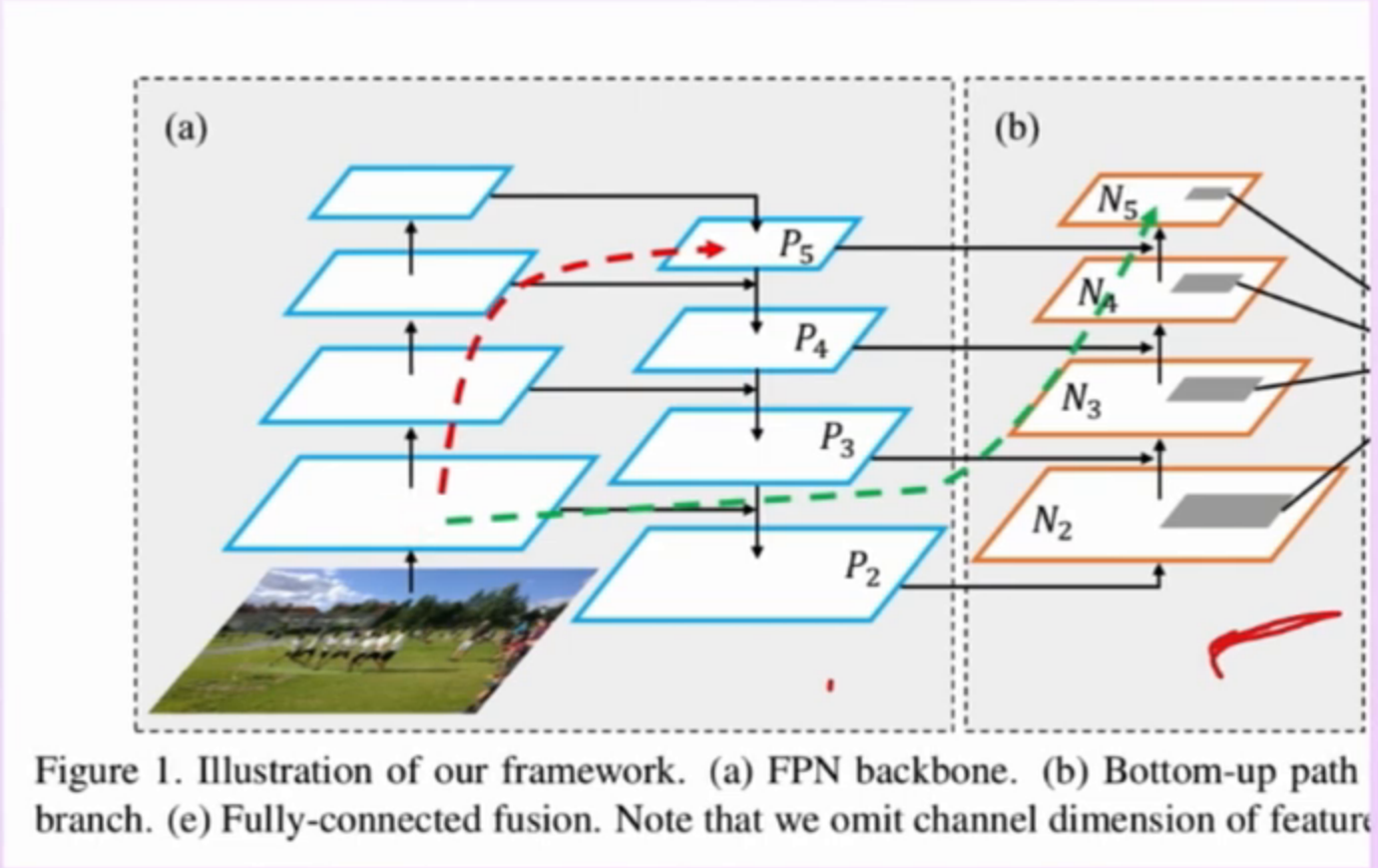
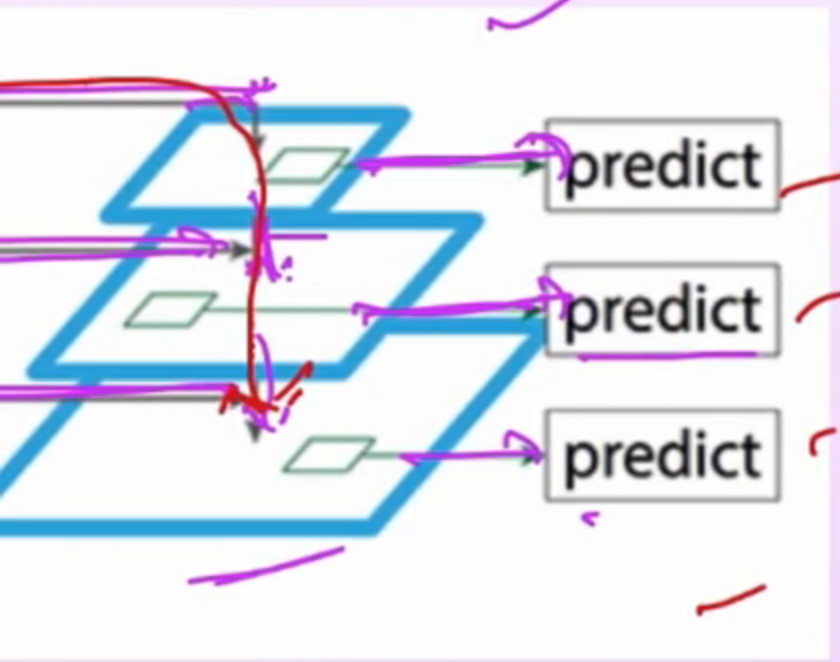


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



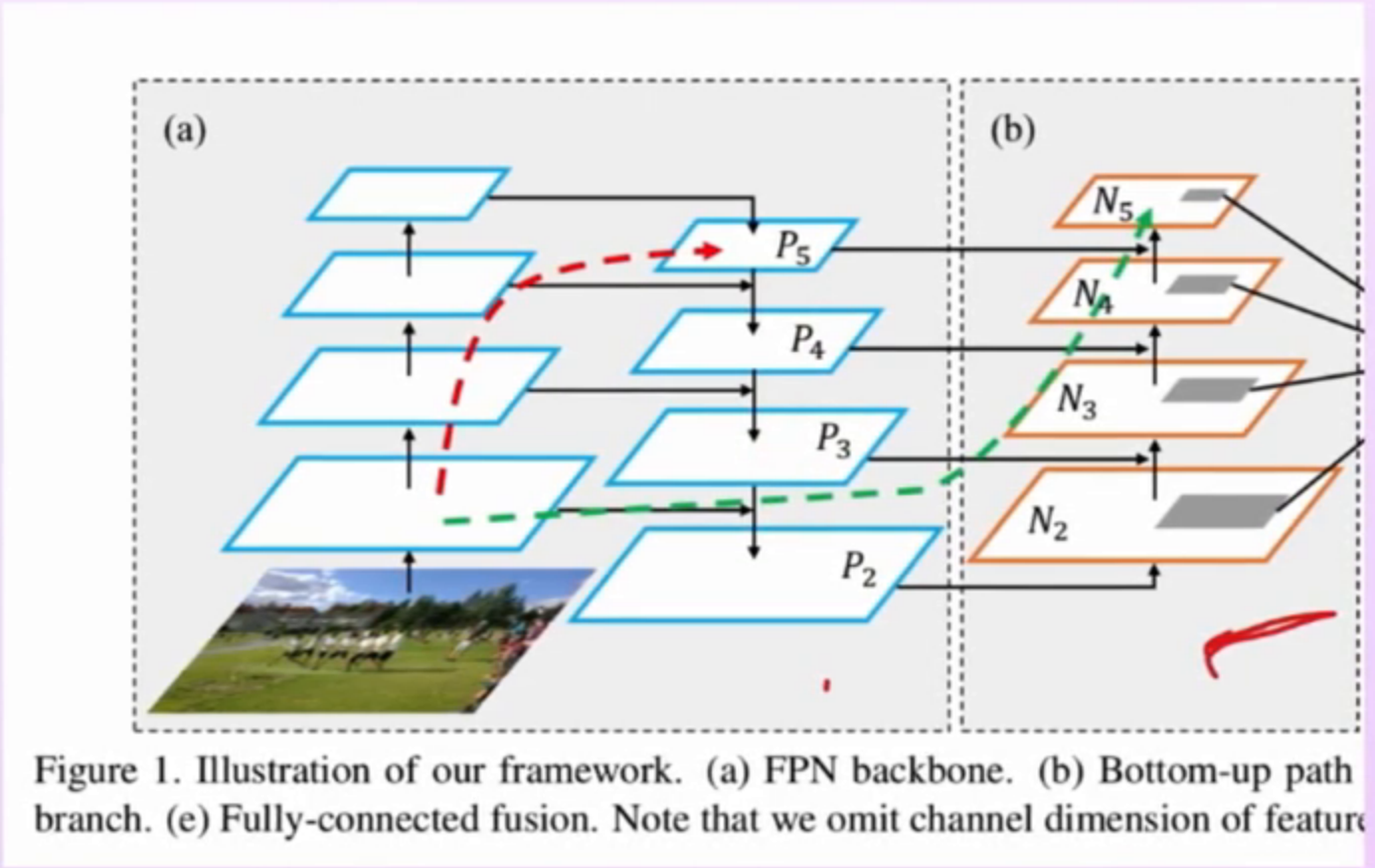
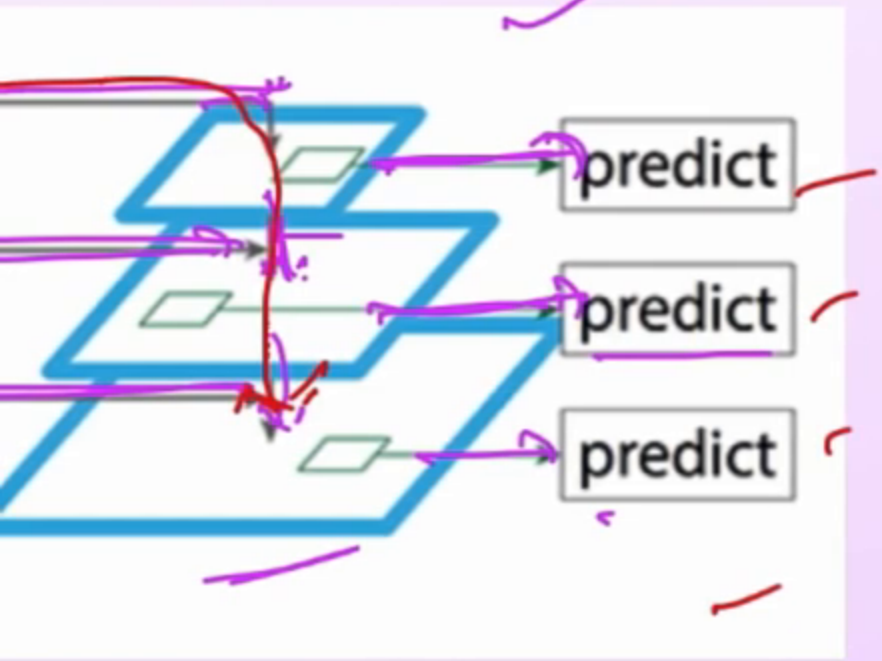


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



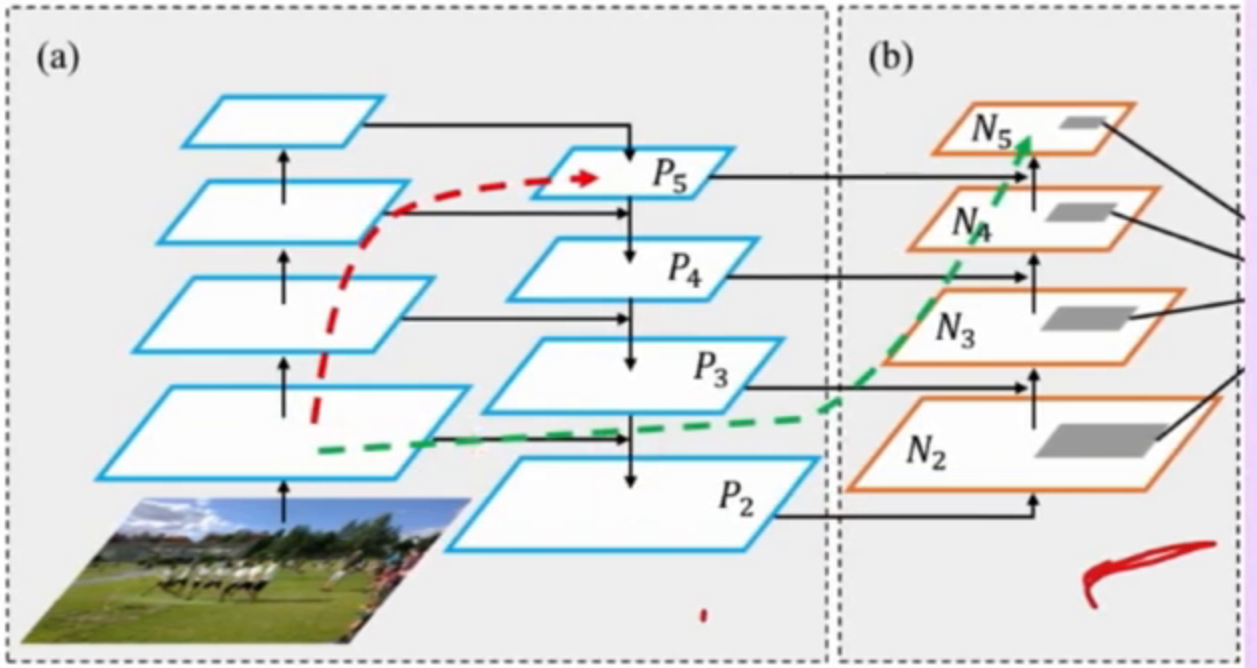
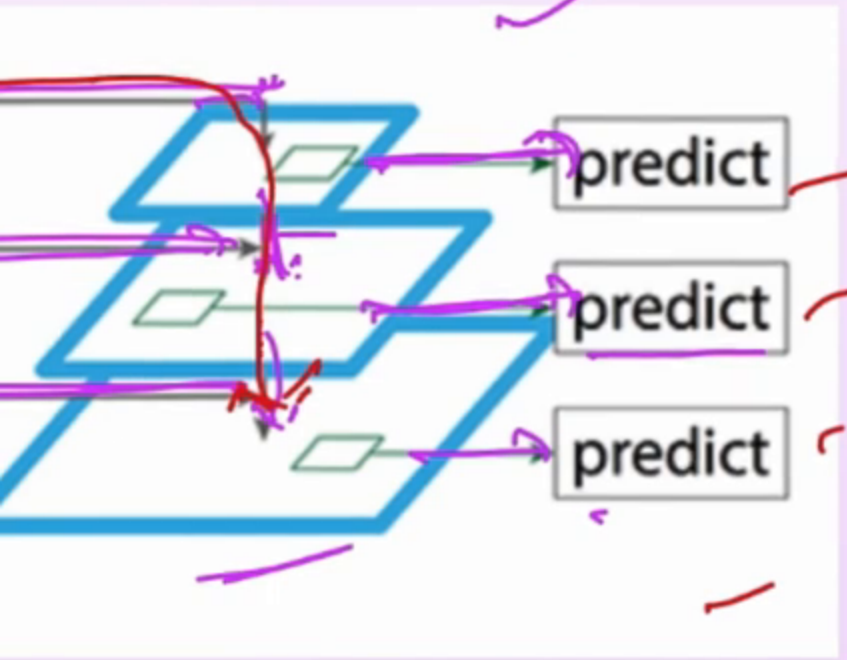


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



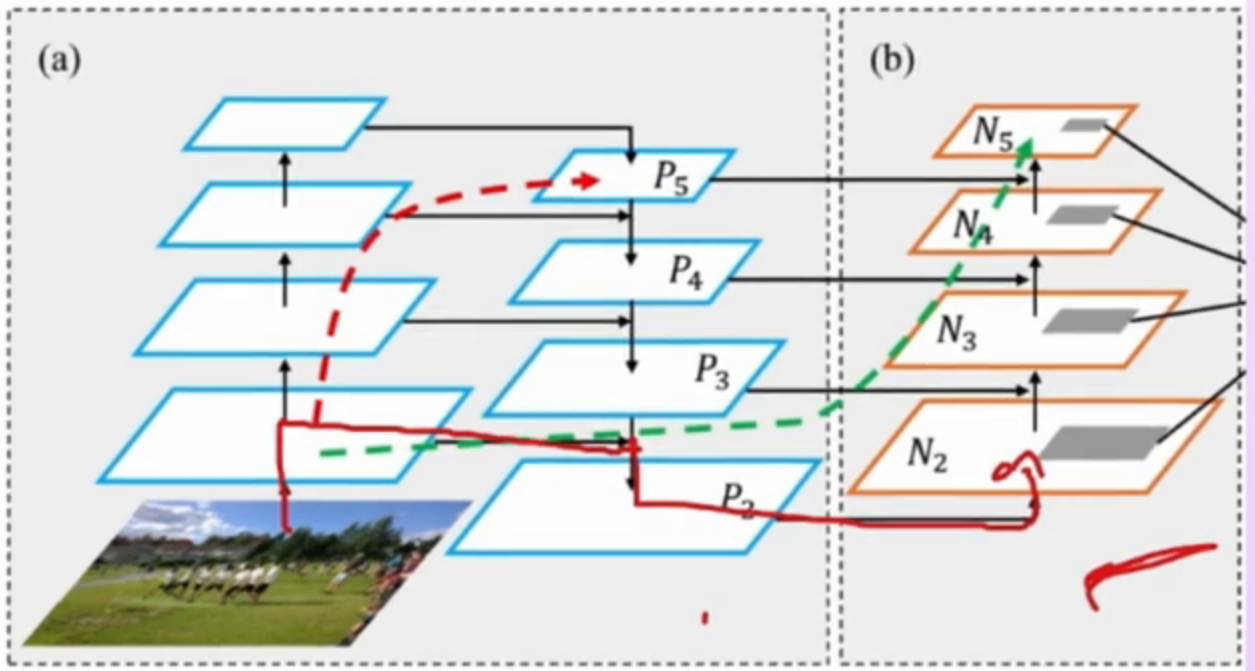
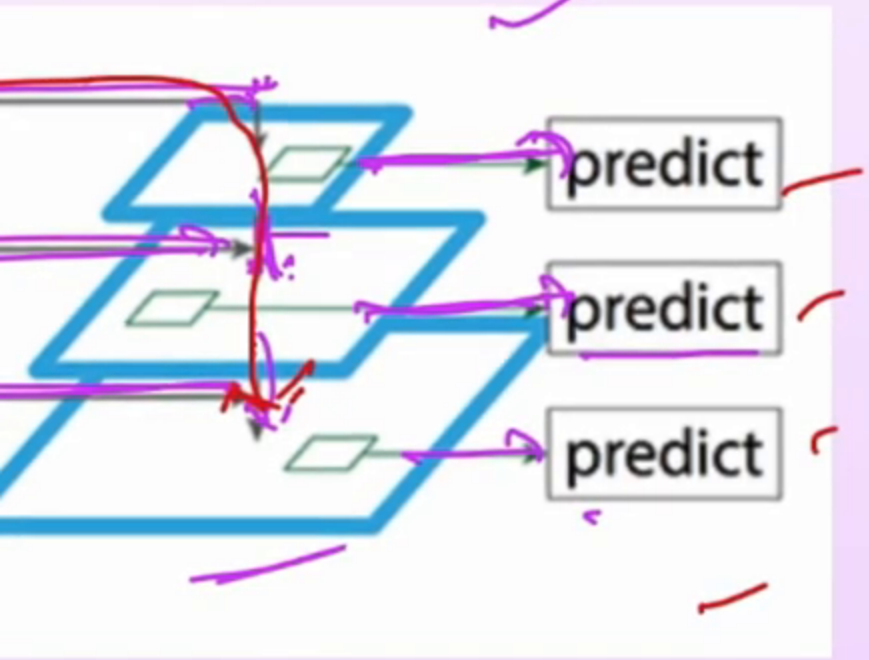


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



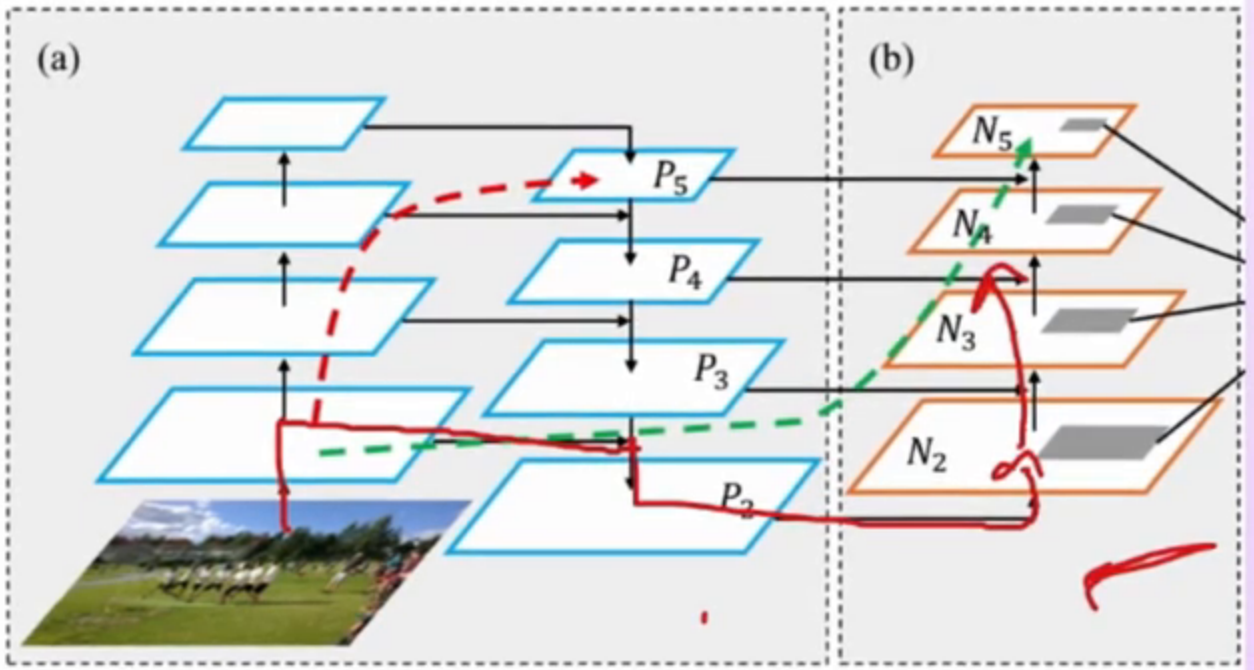
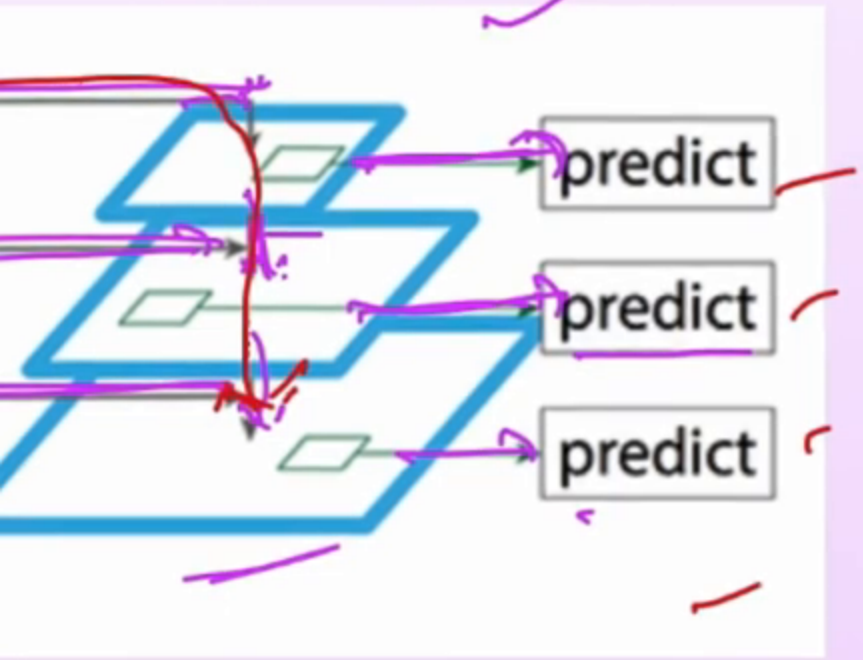


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



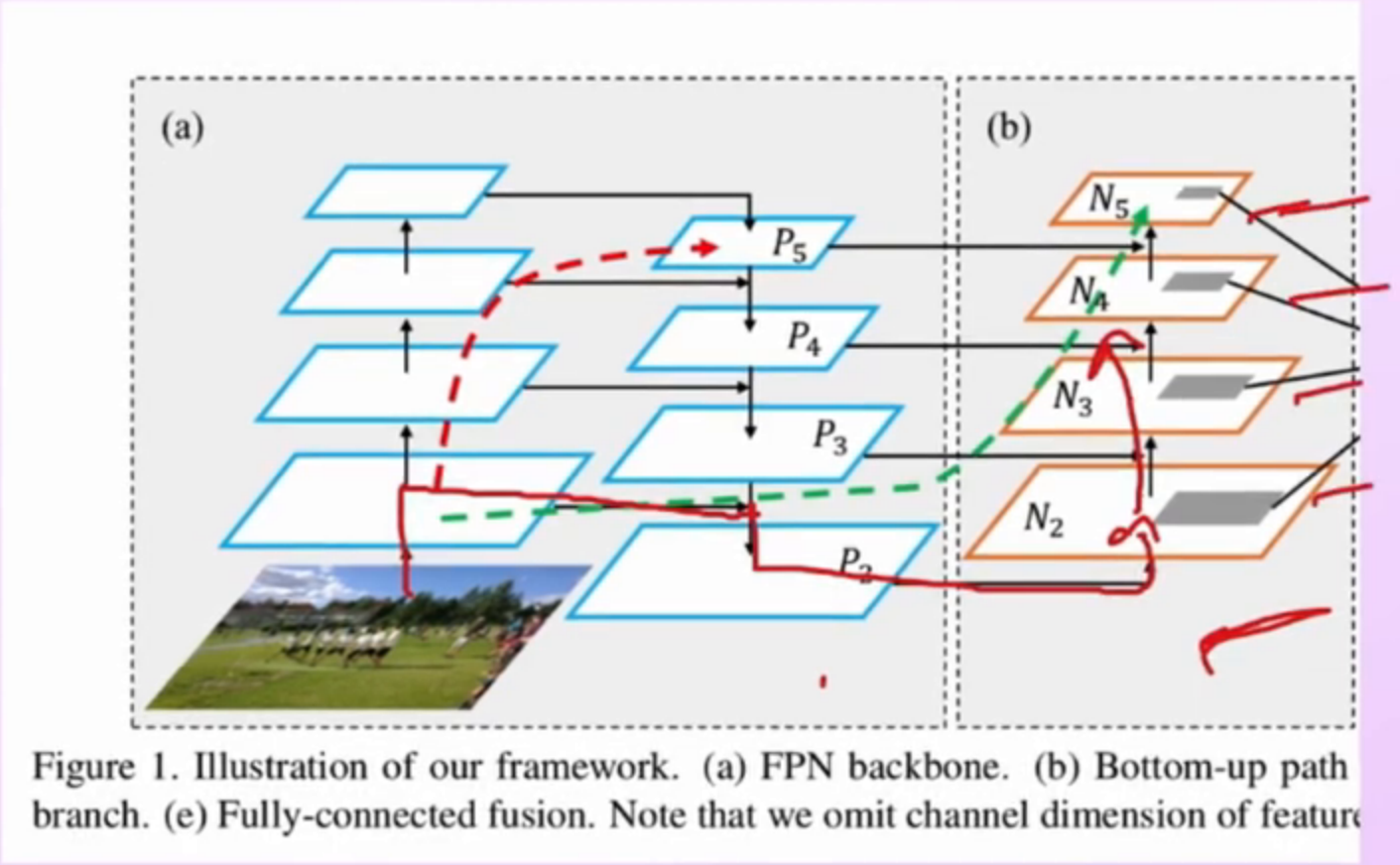
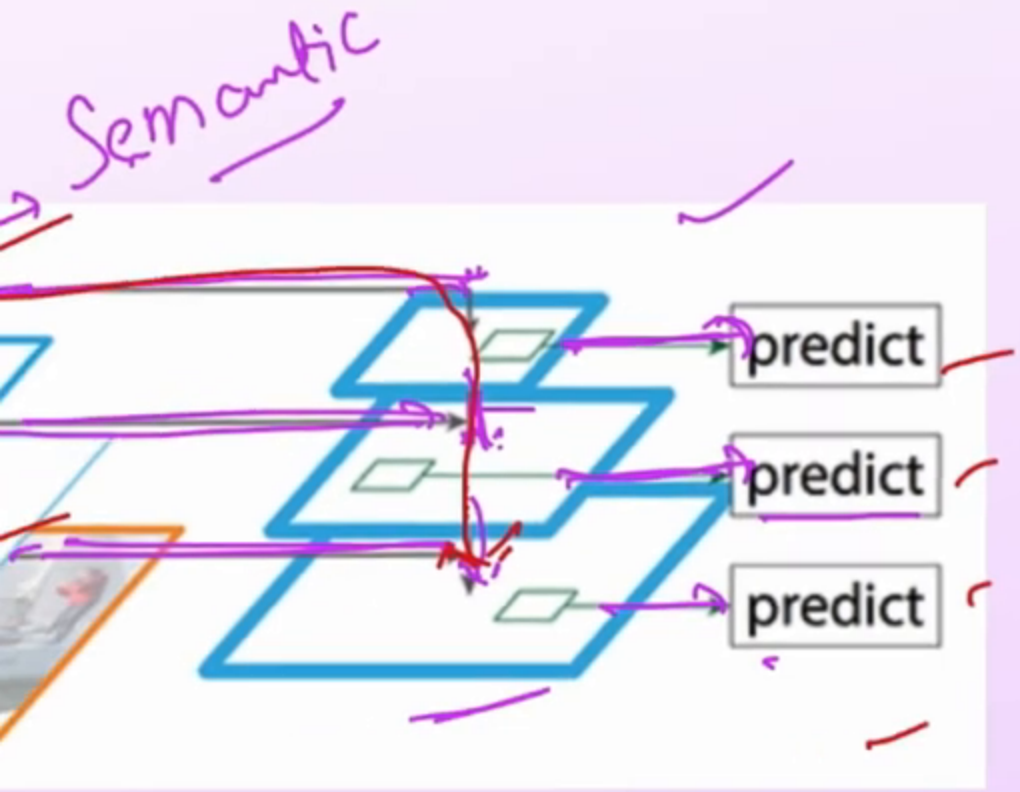


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



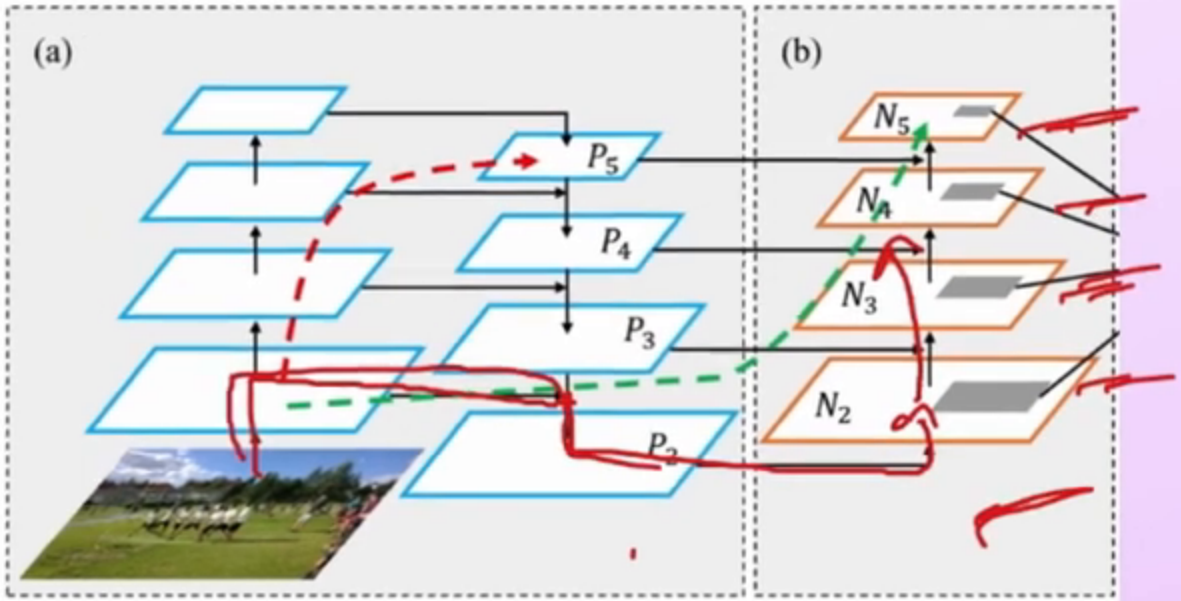
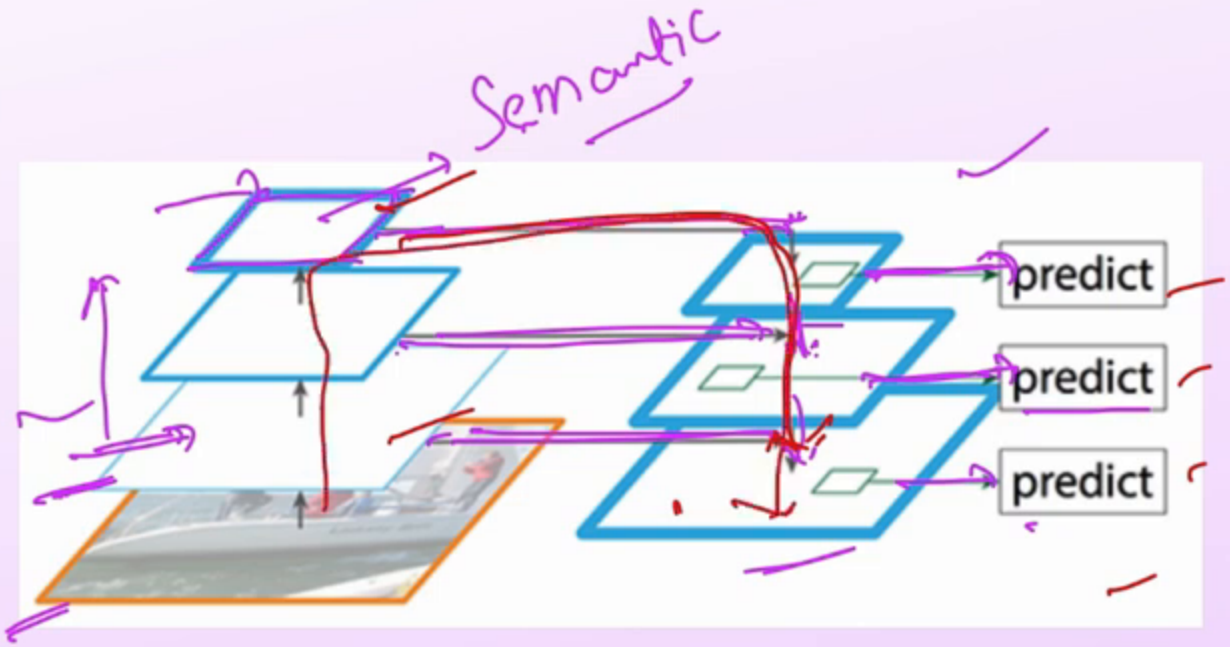


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature

Path Aggregation Network



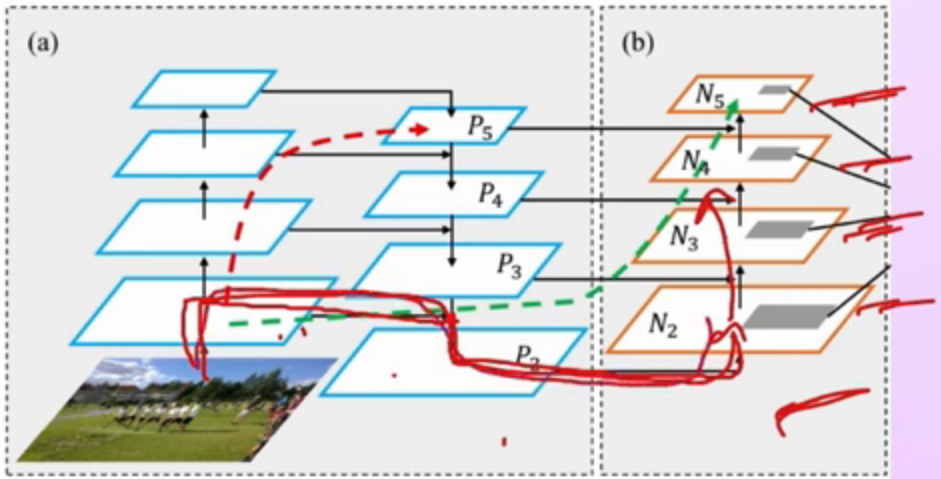
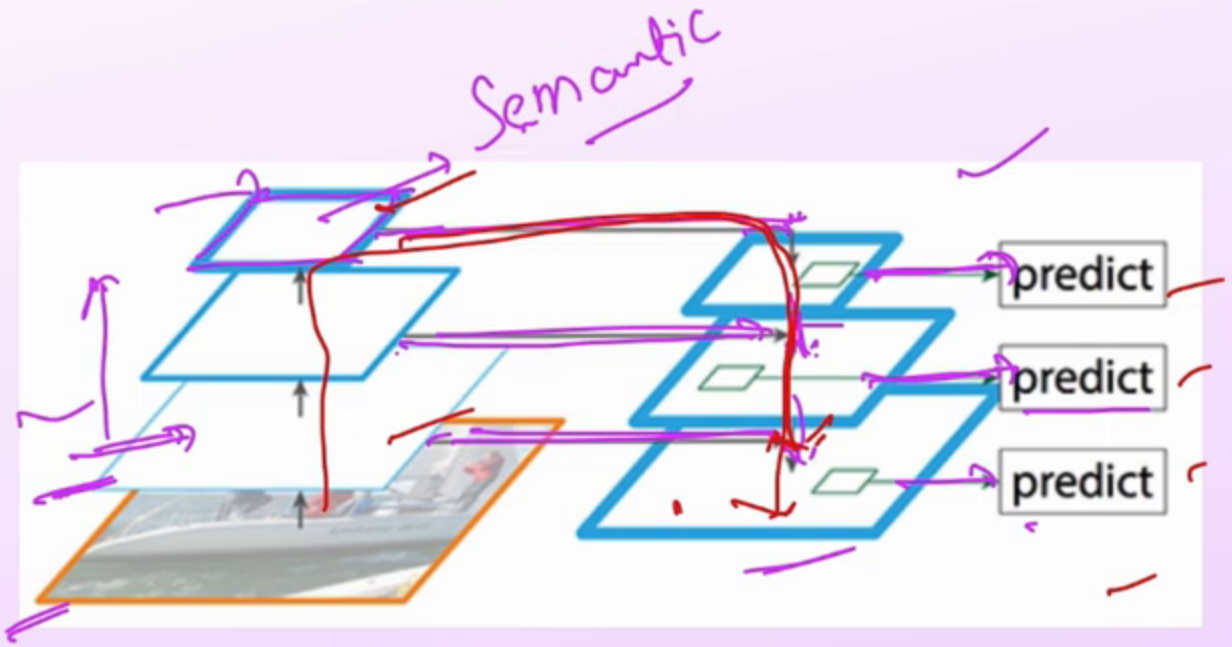


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature

Path Aggregation Network



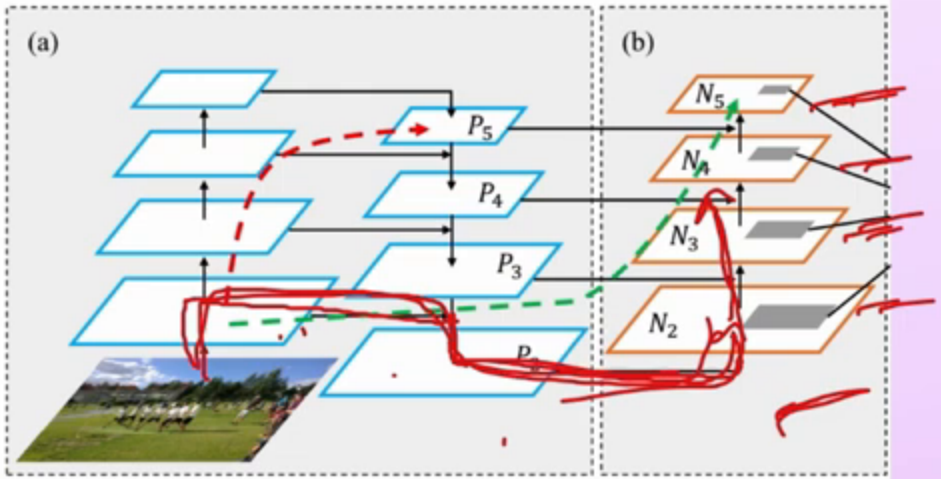


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature

Path Aggregation Network

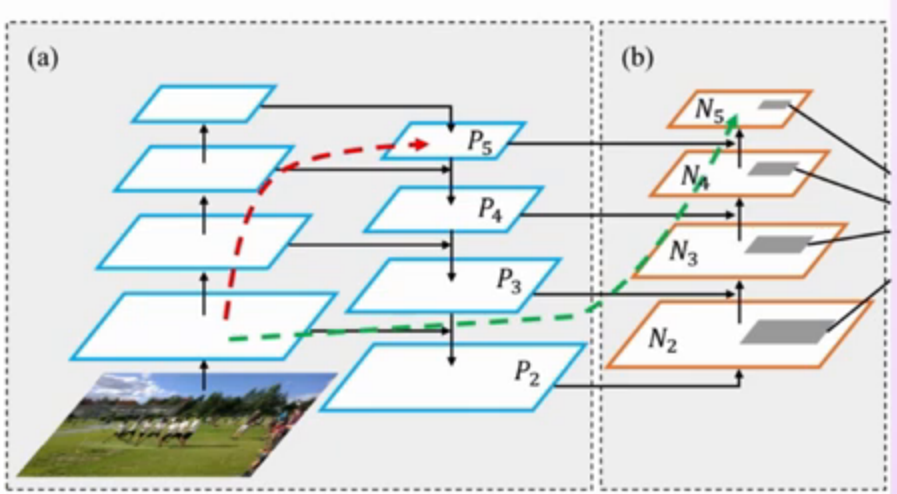
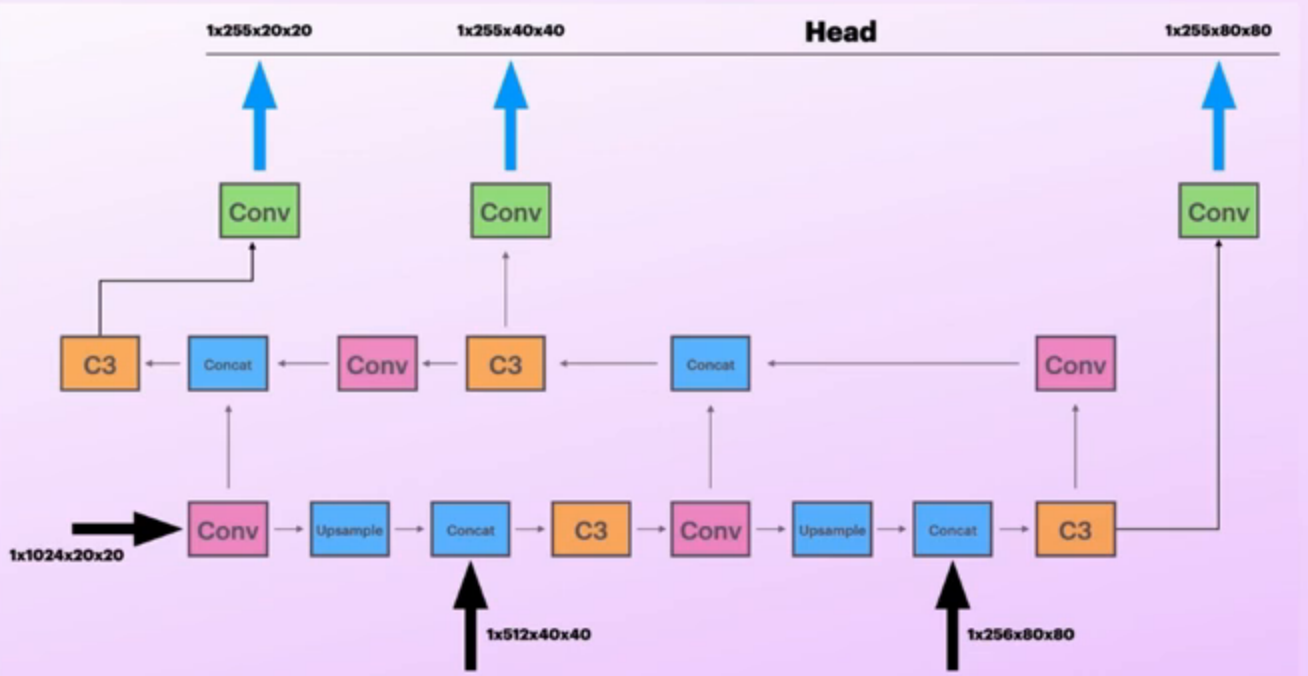


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



Path Aggregation Network

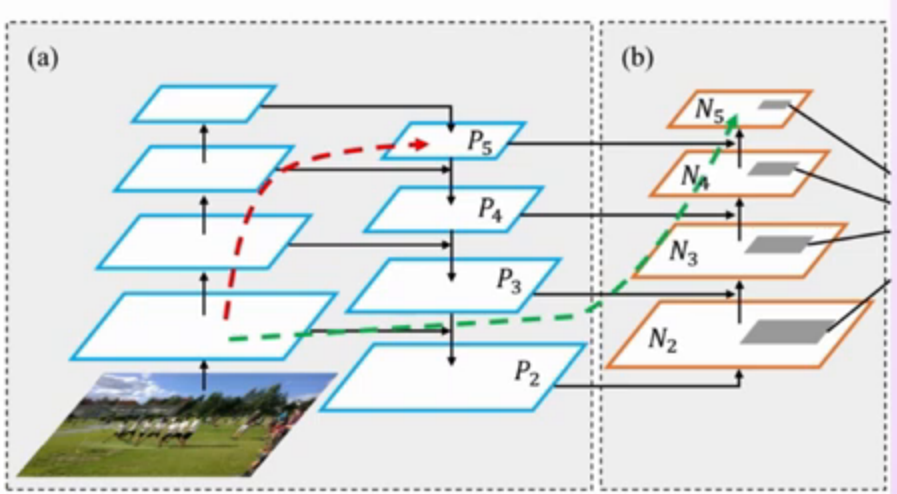
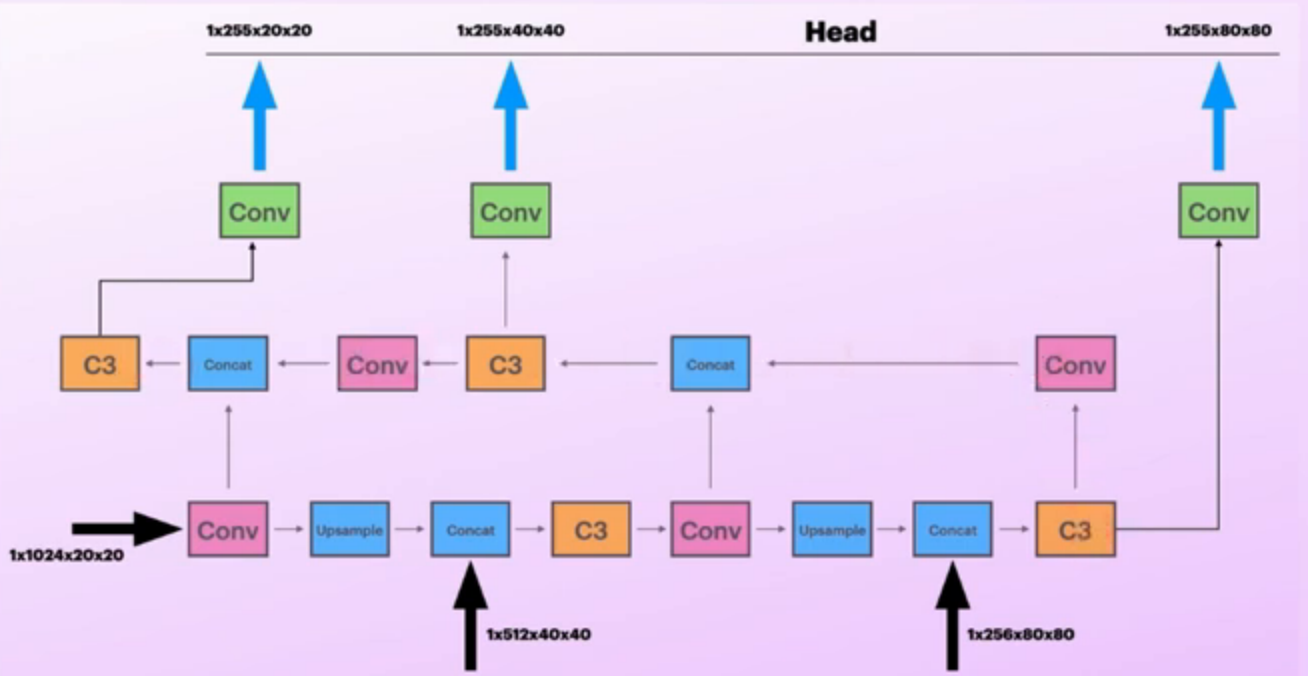


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



Path Aggregation Network

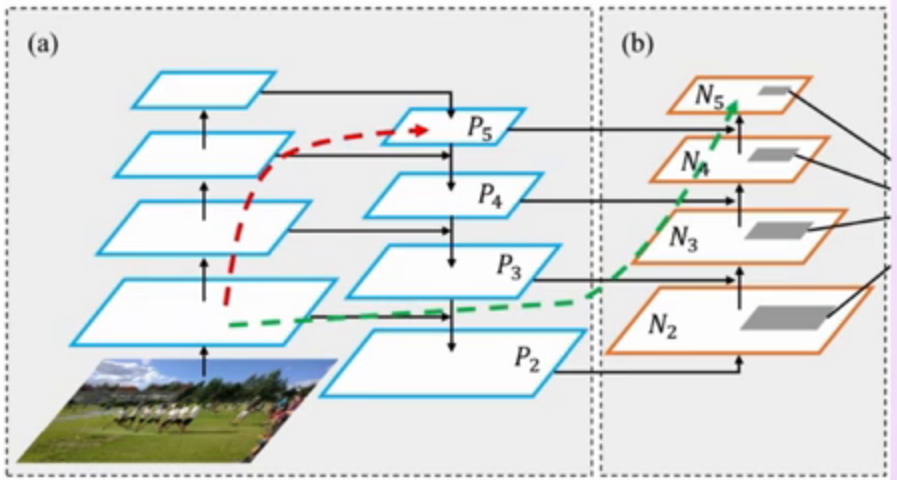
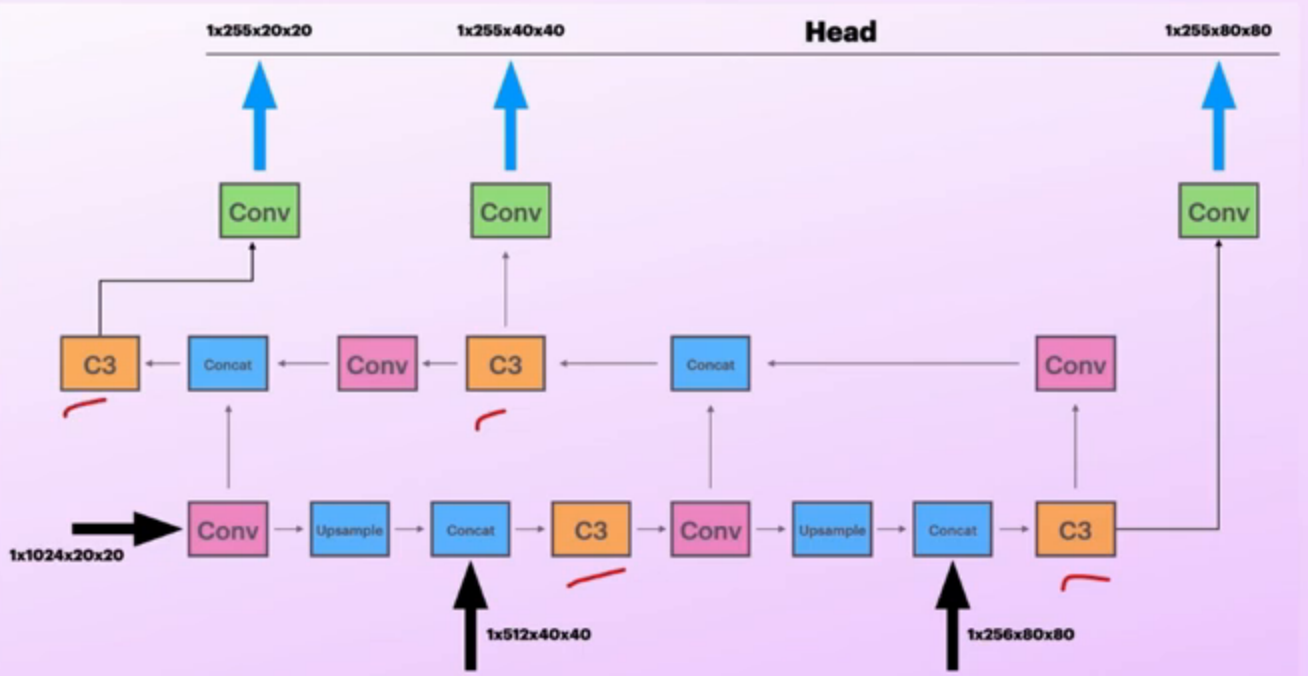


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



Path Aggregation Network

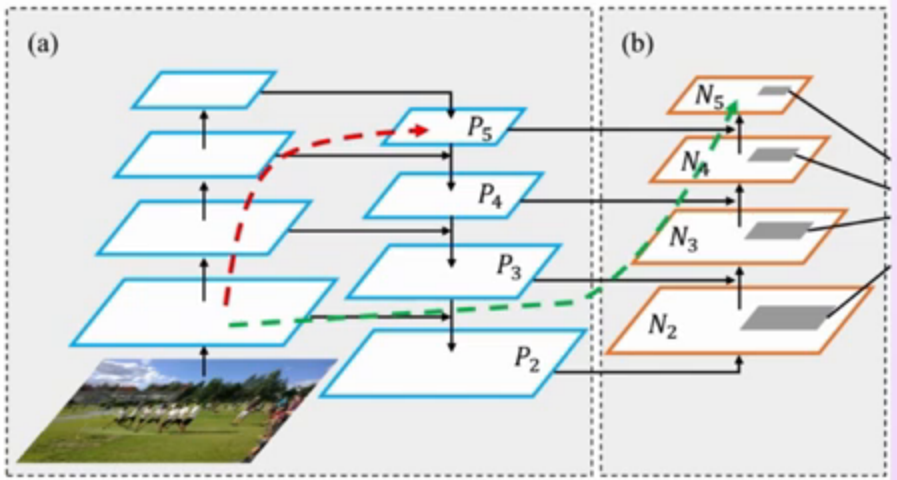
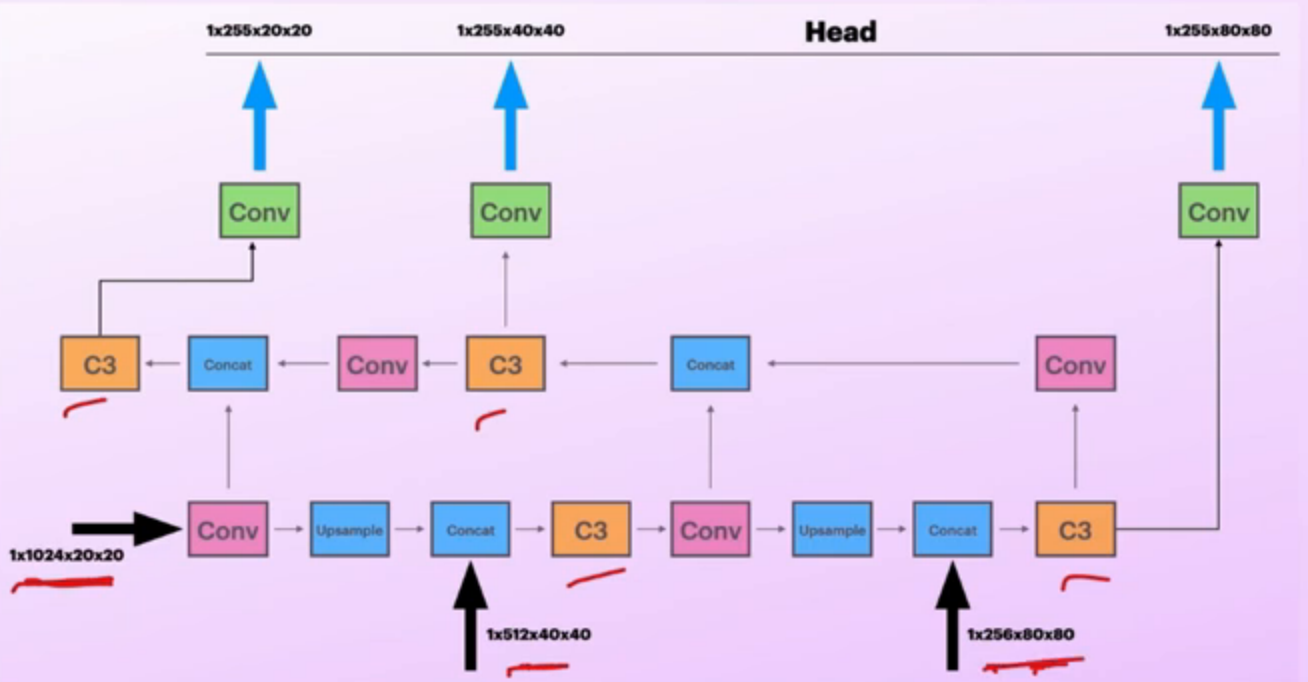


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



Path Aggregation Network

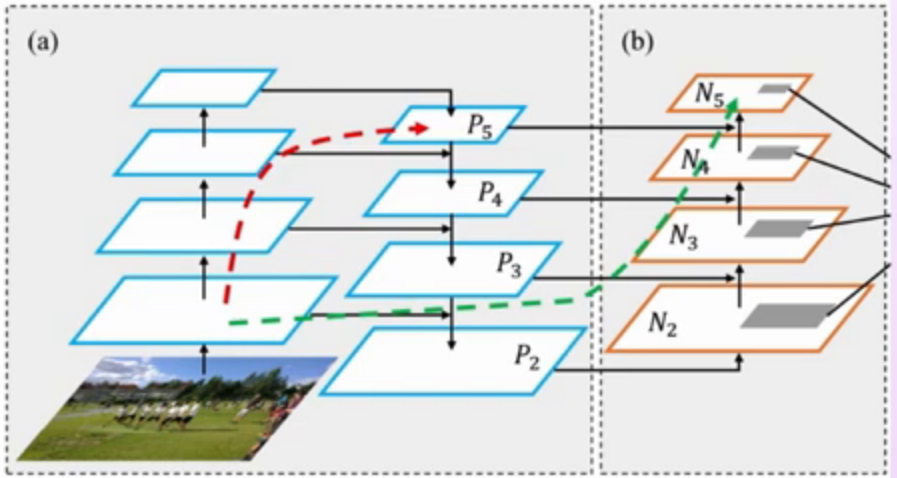
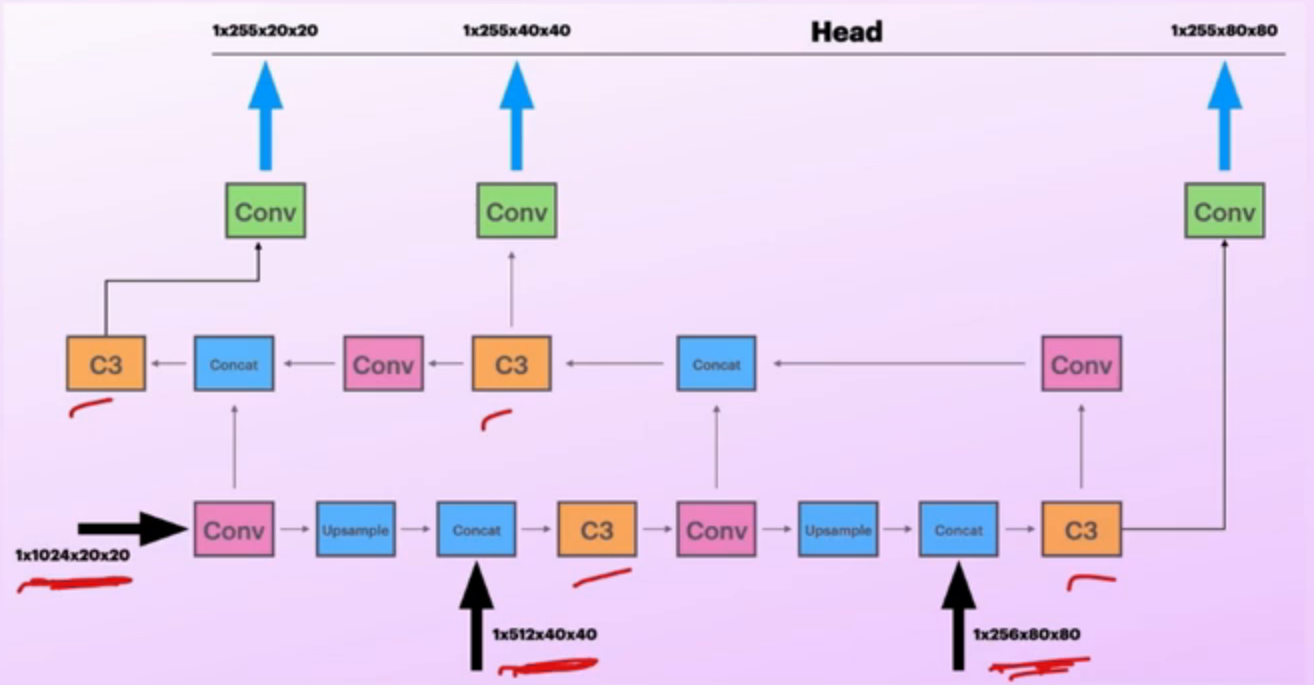


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



Path Aggregation Network

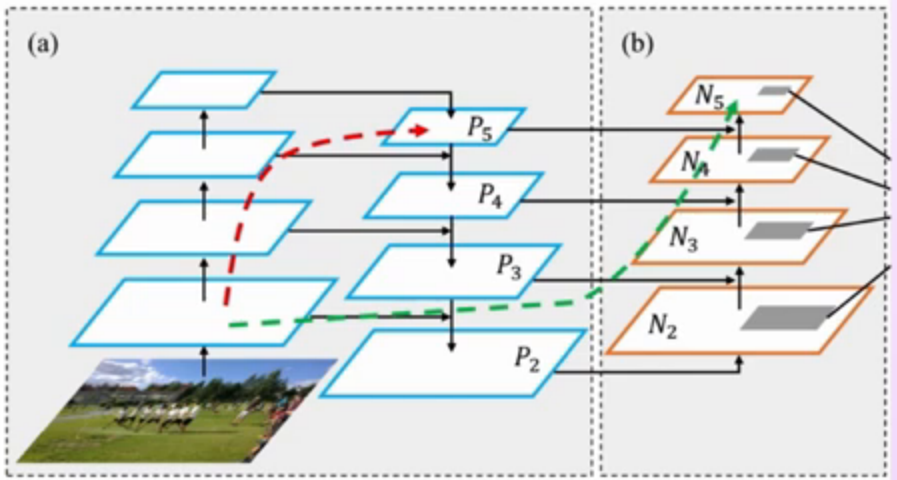
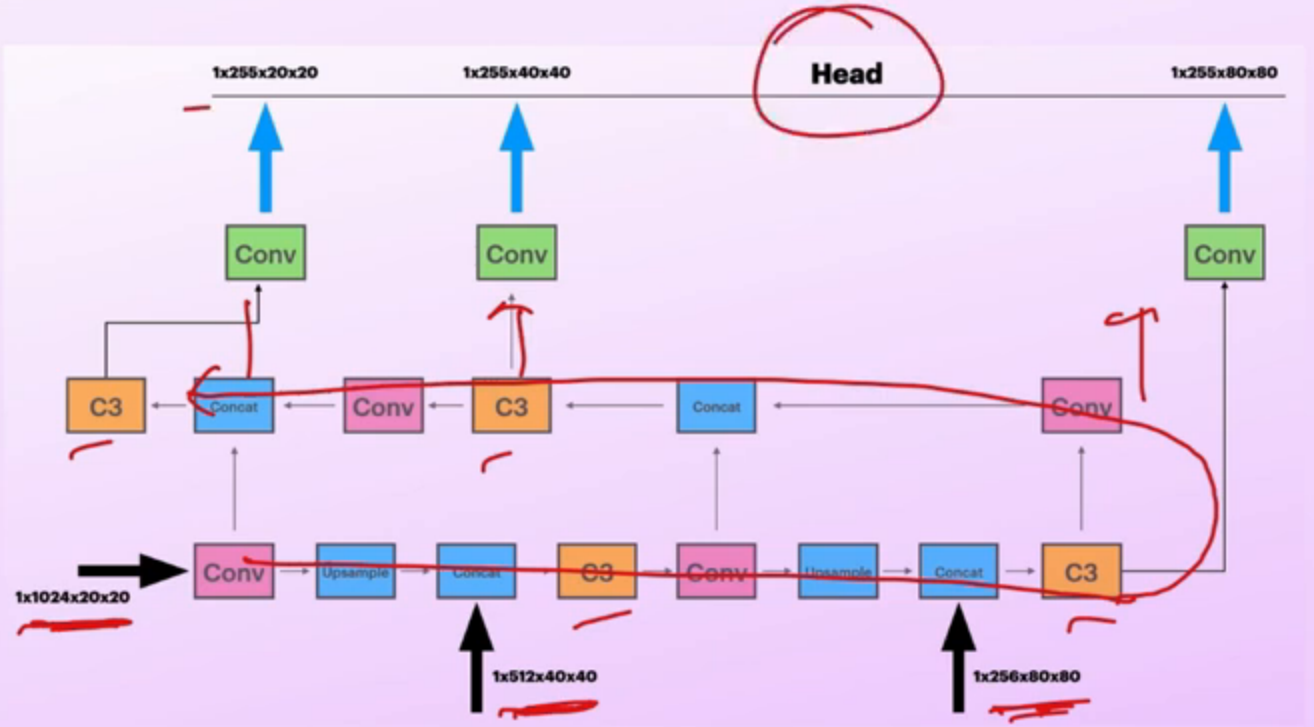


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



Path Aggregation Network

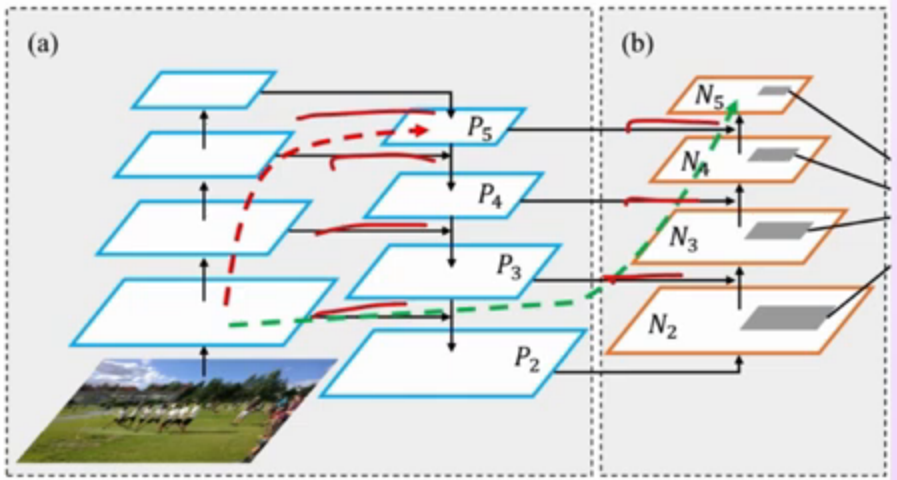
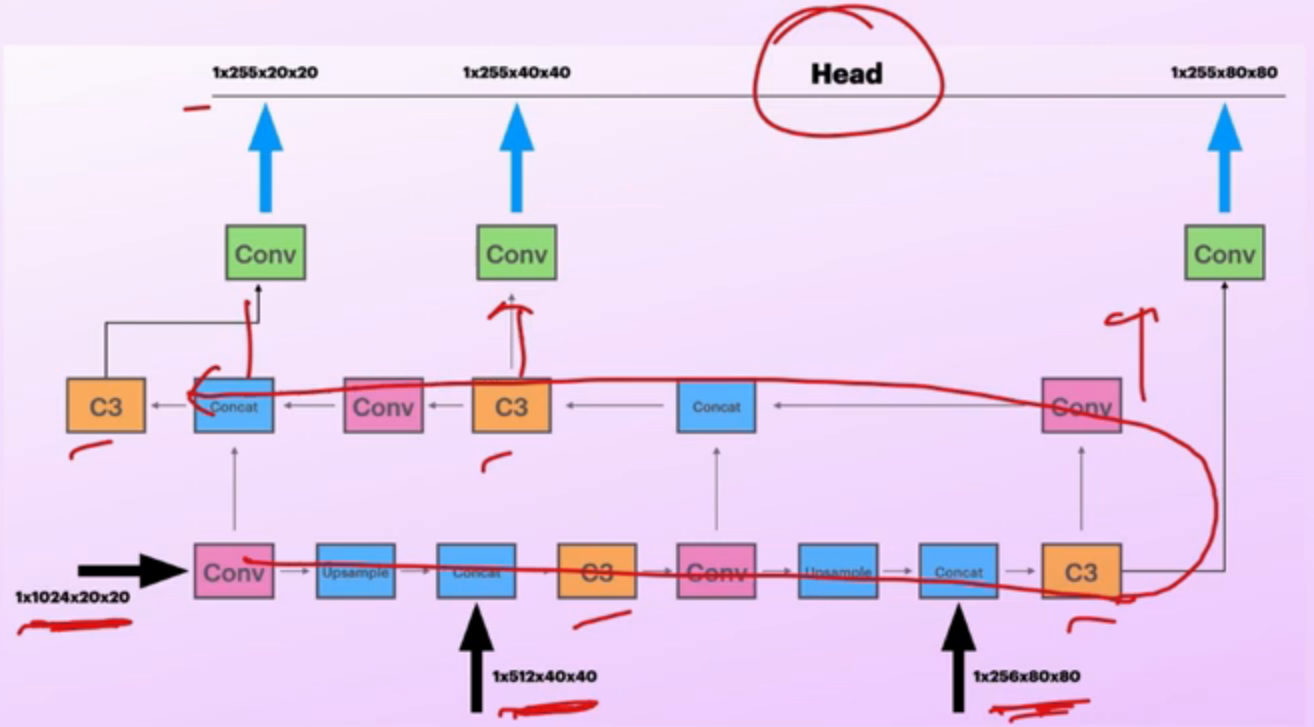
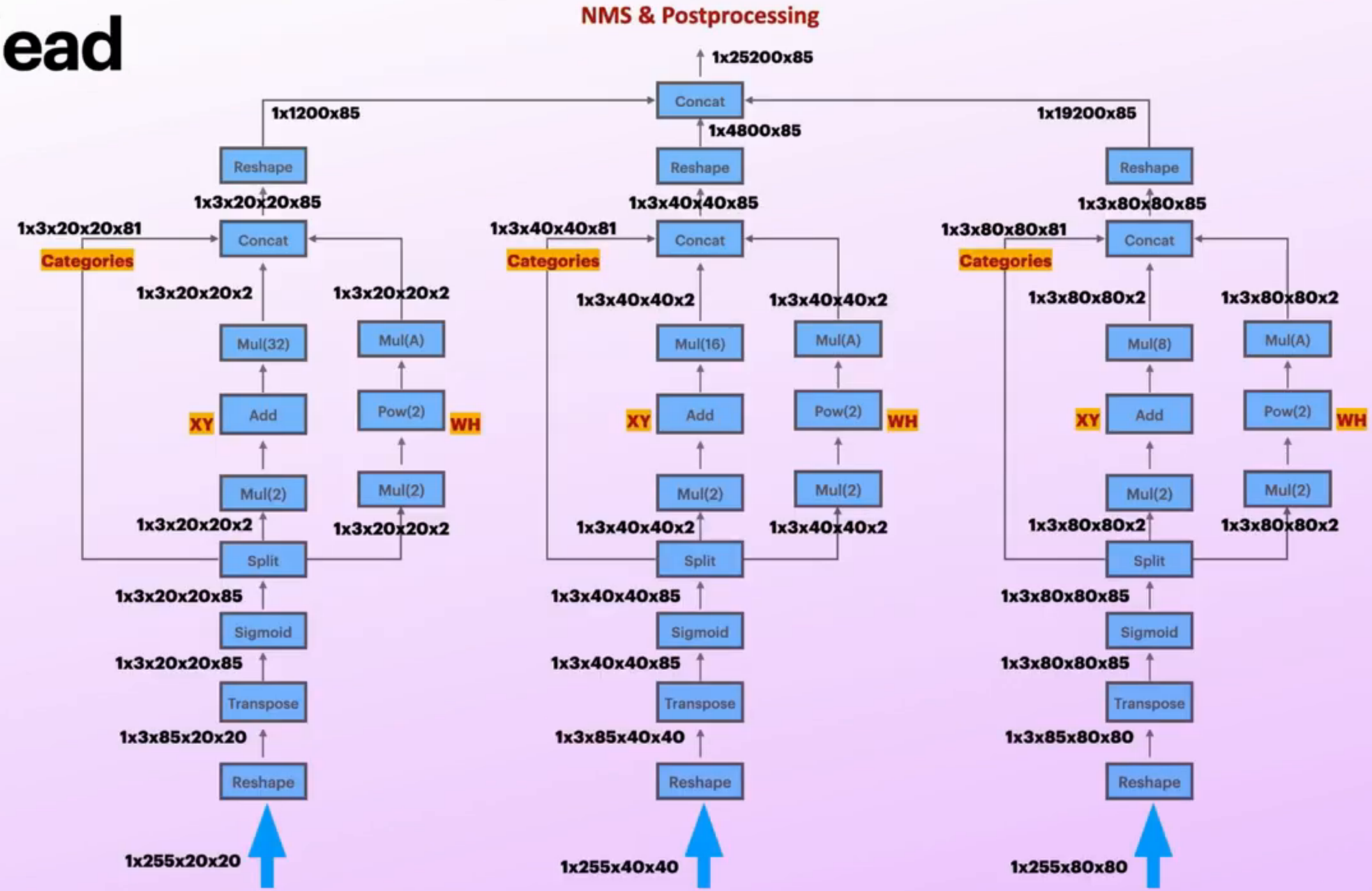


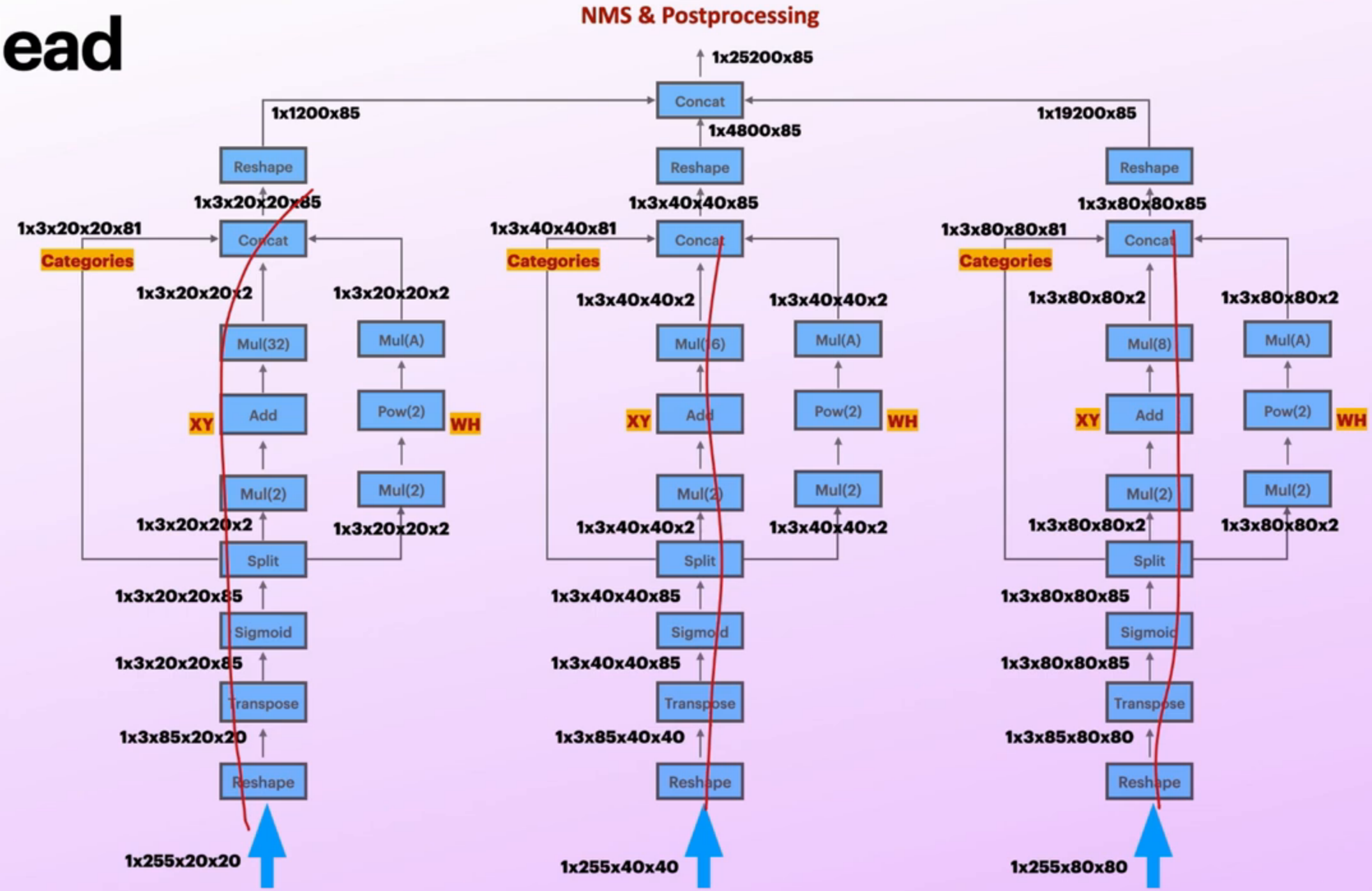
Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path
branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature



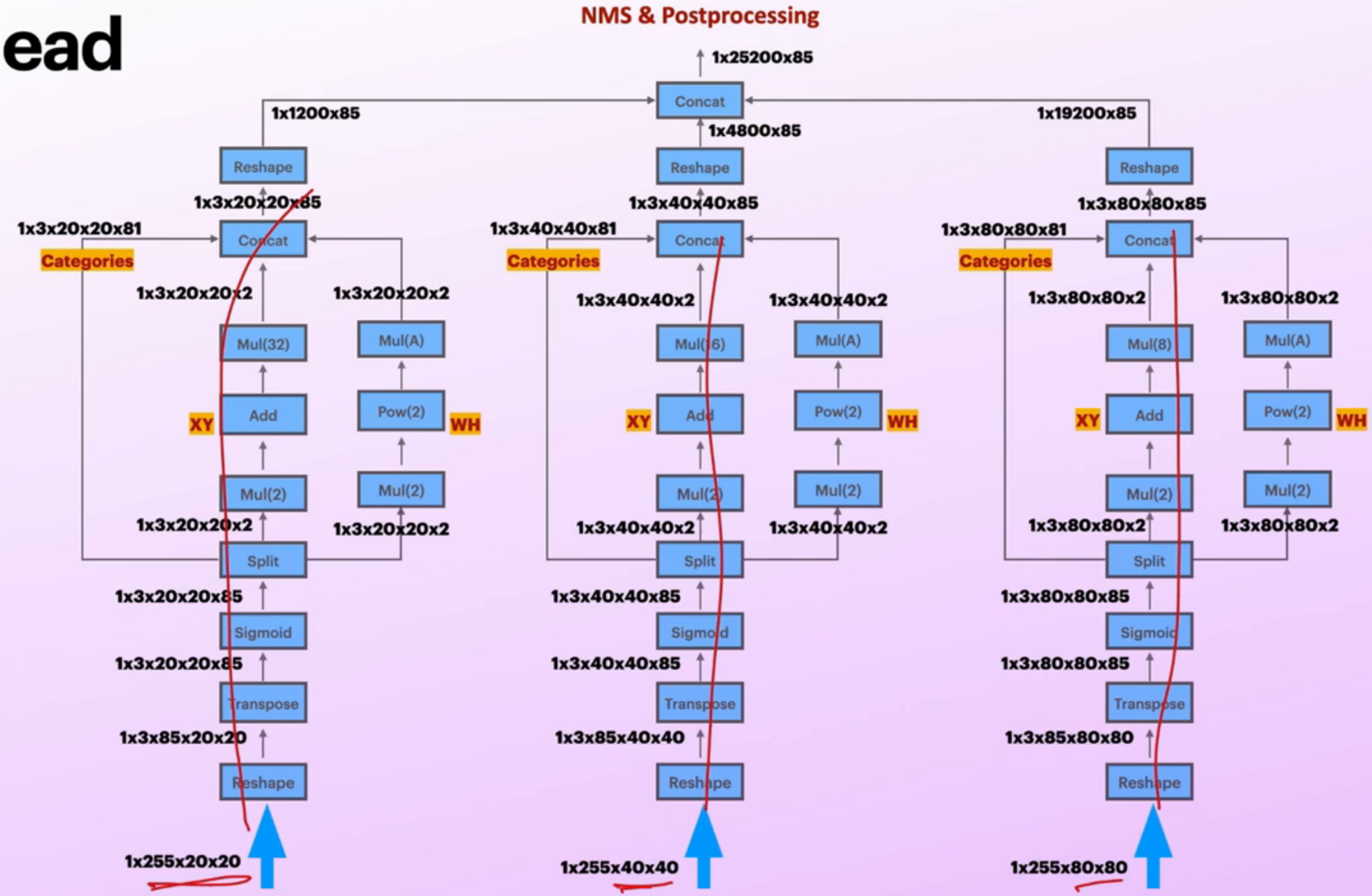
Head

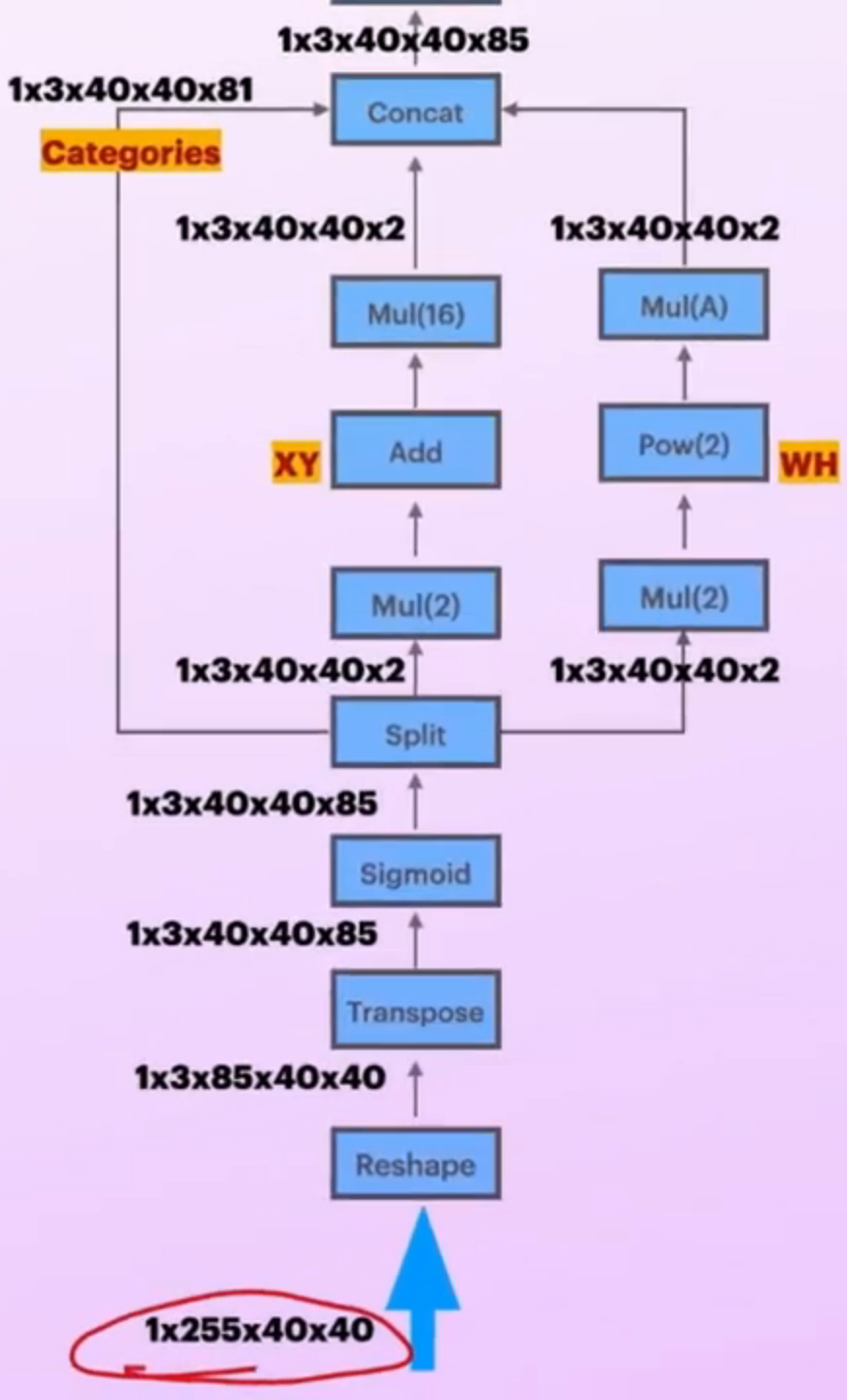

NMS & Postprocessing
1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Reshape
Reshape
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
1x3x20x20x81
1x3x40x40x81
1x3x80x80x81
Concat
Concat
Concat
Categories
Categories
Categories
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Mul(32)
Mul(A)
Mul(16)
Mul(A)
Mul(8)
Mul(A)
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Split
Split
Split
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Sigmoid
Sigmoid
Sigmoid
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Transpose
Transpose
Transpose
1x3x85x20x20
1x3x85x40x40
1x3x85x80x80
Reshape
Reshape
Reshape
1x255x20x20
1x255x40x40
1x255x80x80

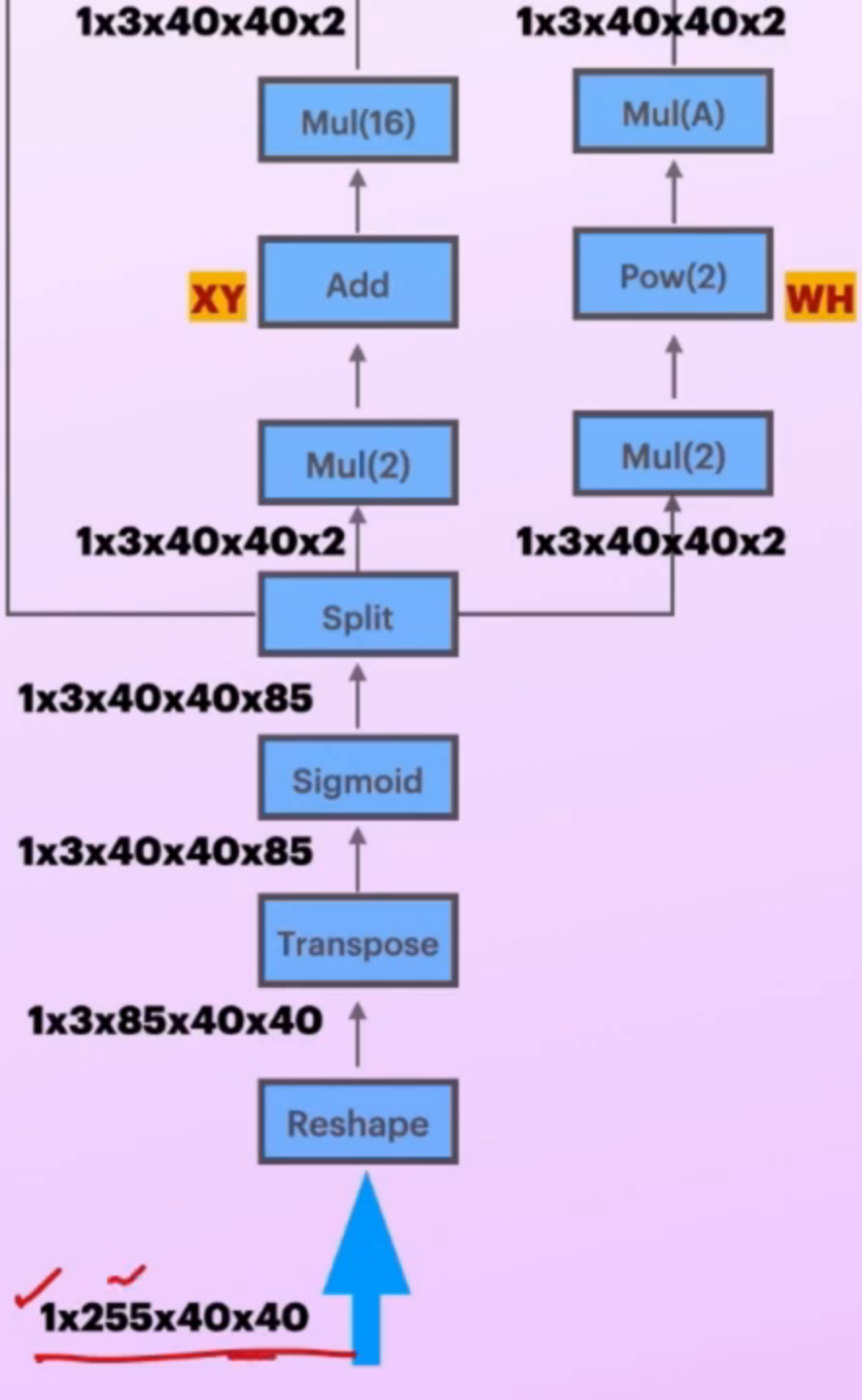
Head

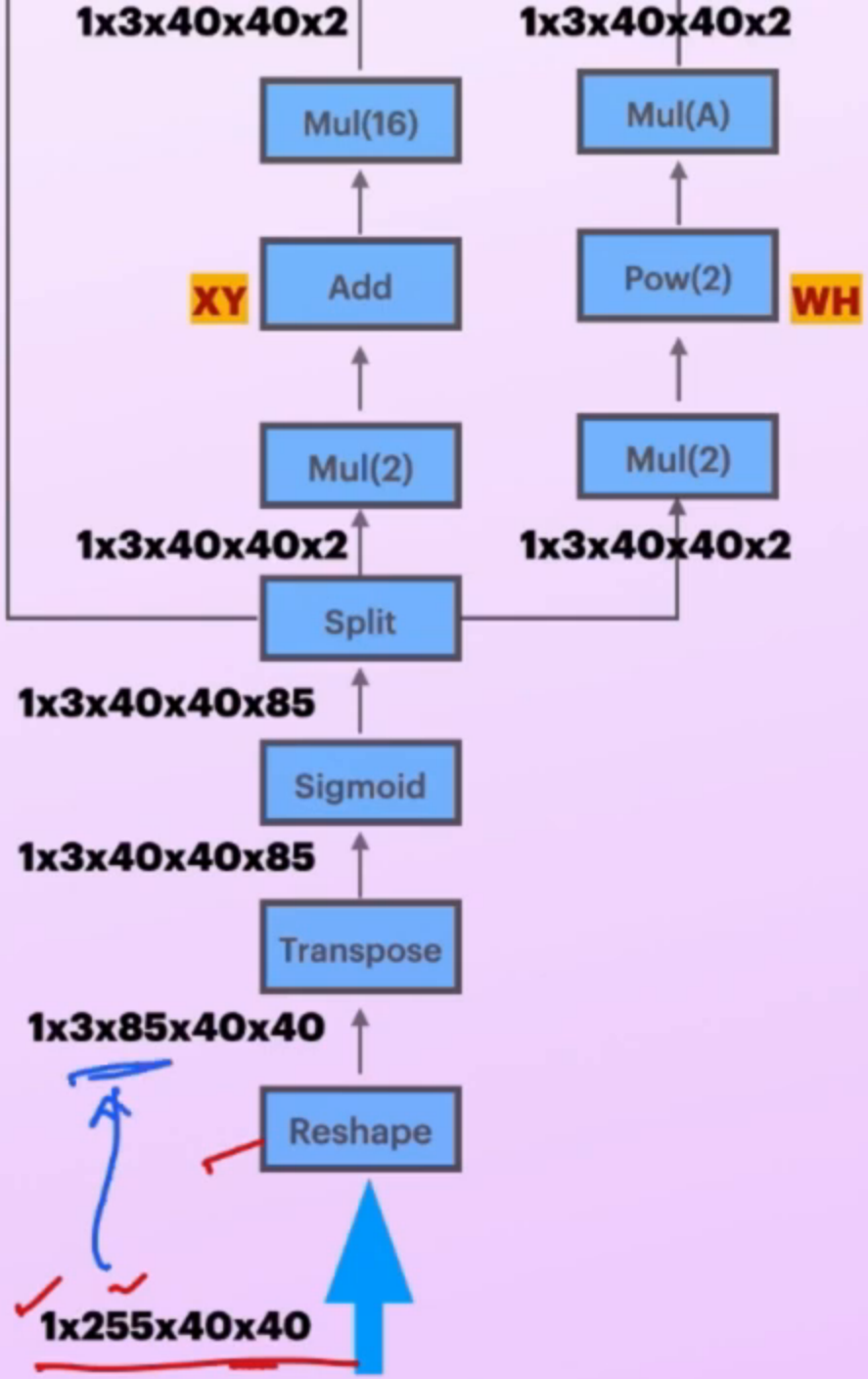

NMS & Postprocessing
1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Reshape
Reshape
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
1x3x20x20x81
1x3x40x40x81
1x3x80x80x81
Concat
Concat
Concat
Categories
Categories
Categories
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Mul(32)
Mul(A)
Mul(16)
Mul(A)
Mul(8)
Mul(A)
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Split
Split
Split
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Sigmoid
Sigmoid
Sigmoid
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Transpose
Transpose
Transpose
1x3x85x20x20
1x3x85x40x40
1x3x85x80x80
Reshape
Reshape
Reshape
1x255x20x20
1x255x40x40
1x255x80x80

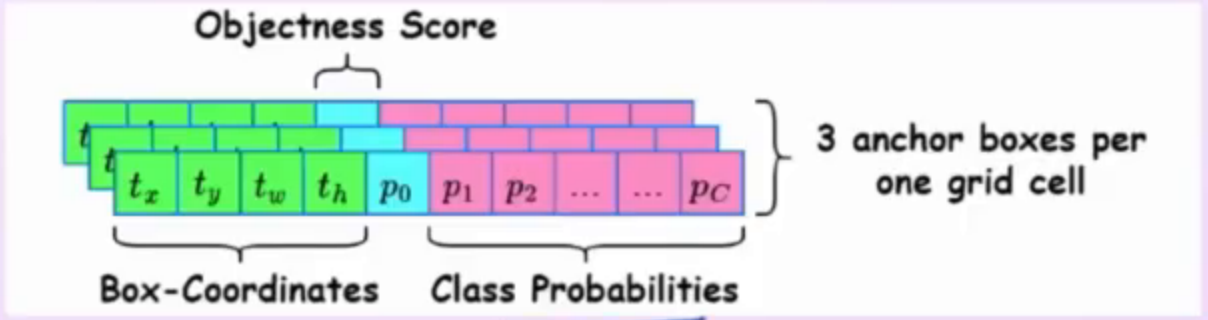
Head

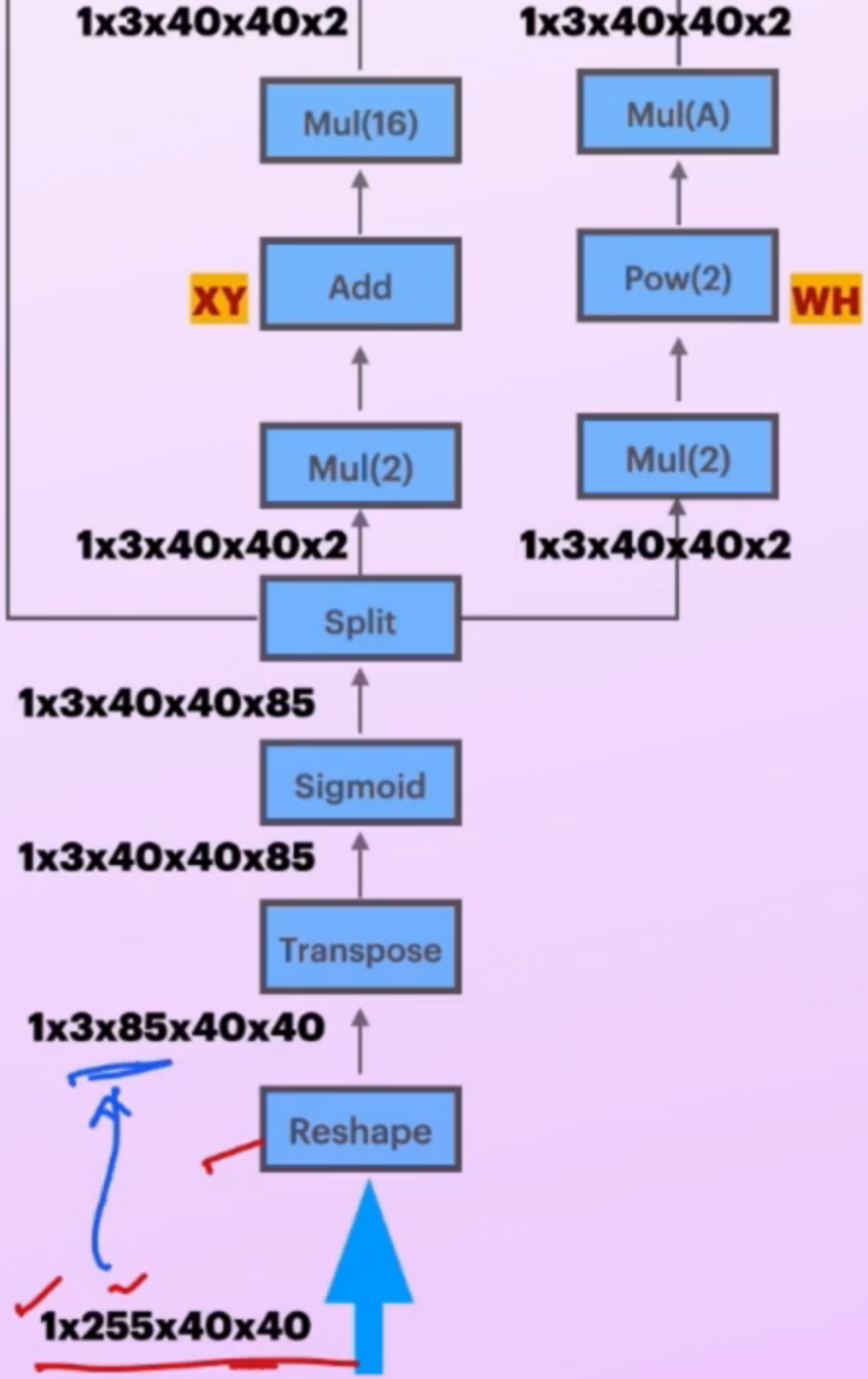

NMS & Postprocessing
1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Reshape
Reshape
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
1x3x20x20x81
1x3x40x40x81
1x3x80x80x81
Concat
Concat
Concat
Categories
Categories
Categories
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Mul(32)
Mul(A)
Mul(16)
Mul(A)
Mul(8)
Mul(A)
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Split
Split
Split
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Sigmoid
Sigmoid
Sigmoid
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Transpose
Transpose
Transpose
1x3x85x20x20
1x3x85x40x40
1x3x85x80x80
Reshape
Reshape
Reshape
1x255x20x20
1x255x40x40
1x255x80x80

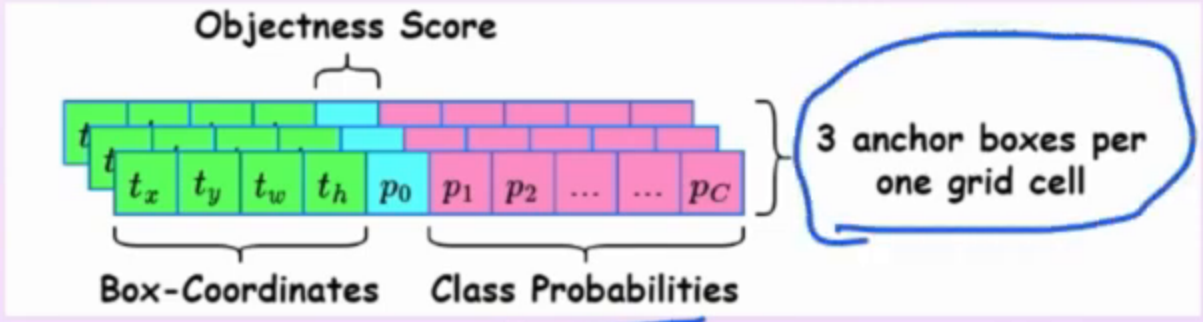

1x3x40x40x85
1x3x40x40x81
Concat
Categories
1x3x40x40x2
1x3x40x40x2
Mul(16)
Mul(A)
Add
Pow(2)
XY
WH
Mul(2)
Mul(2)
1x3x40x40x2
1x3x40x40x2
Split
1x3x40x40x85
Sigmoid
1x3x40x40x85
Transpose
1x3x85x40x40
Reshape
1x255x40x40

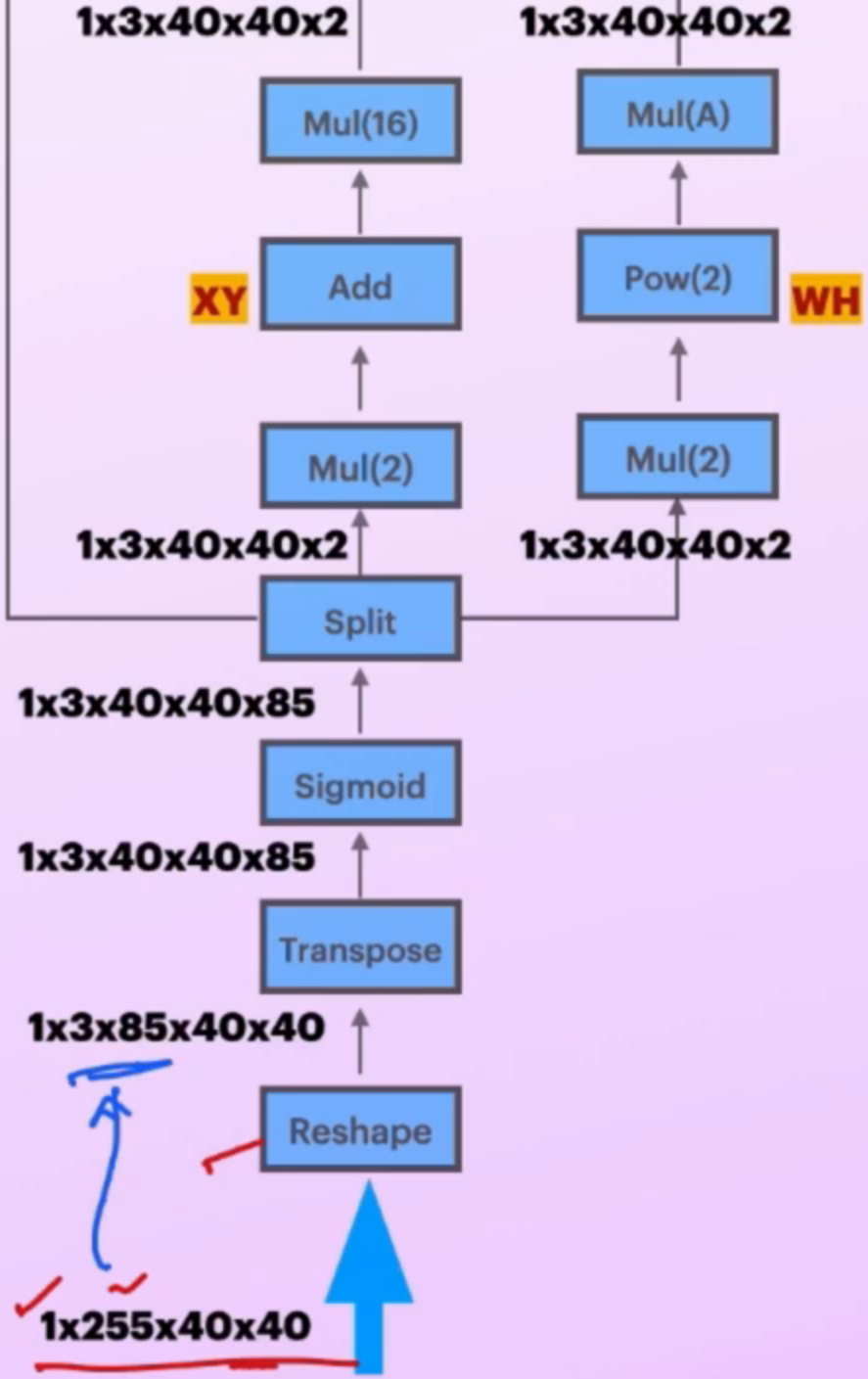

1x3x40x40x2
1x3x40x40x2
Mul(16)
Mul(A)
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
1x3x40x40x2
1x3x40x40x2
Split
1x3x40x40x85
Sigmoid
1x3x40x40x85
Transpose
1x3x85x40x40
Reshape
1x255x40x40

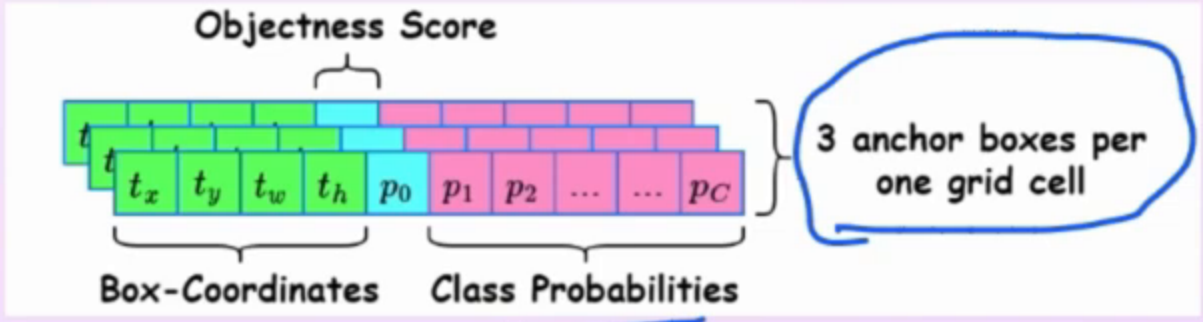

1x3x40x40x2
1x3x40x40x2
Mul(16)
Mul(A)
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
1x3x40x40x2
1x3x40x40x2
Split
1x3x40x40x85
Sigmoid
1x3x40x40x85
Transpose
1x3x85x40x40
Reshape
1x255x40x40

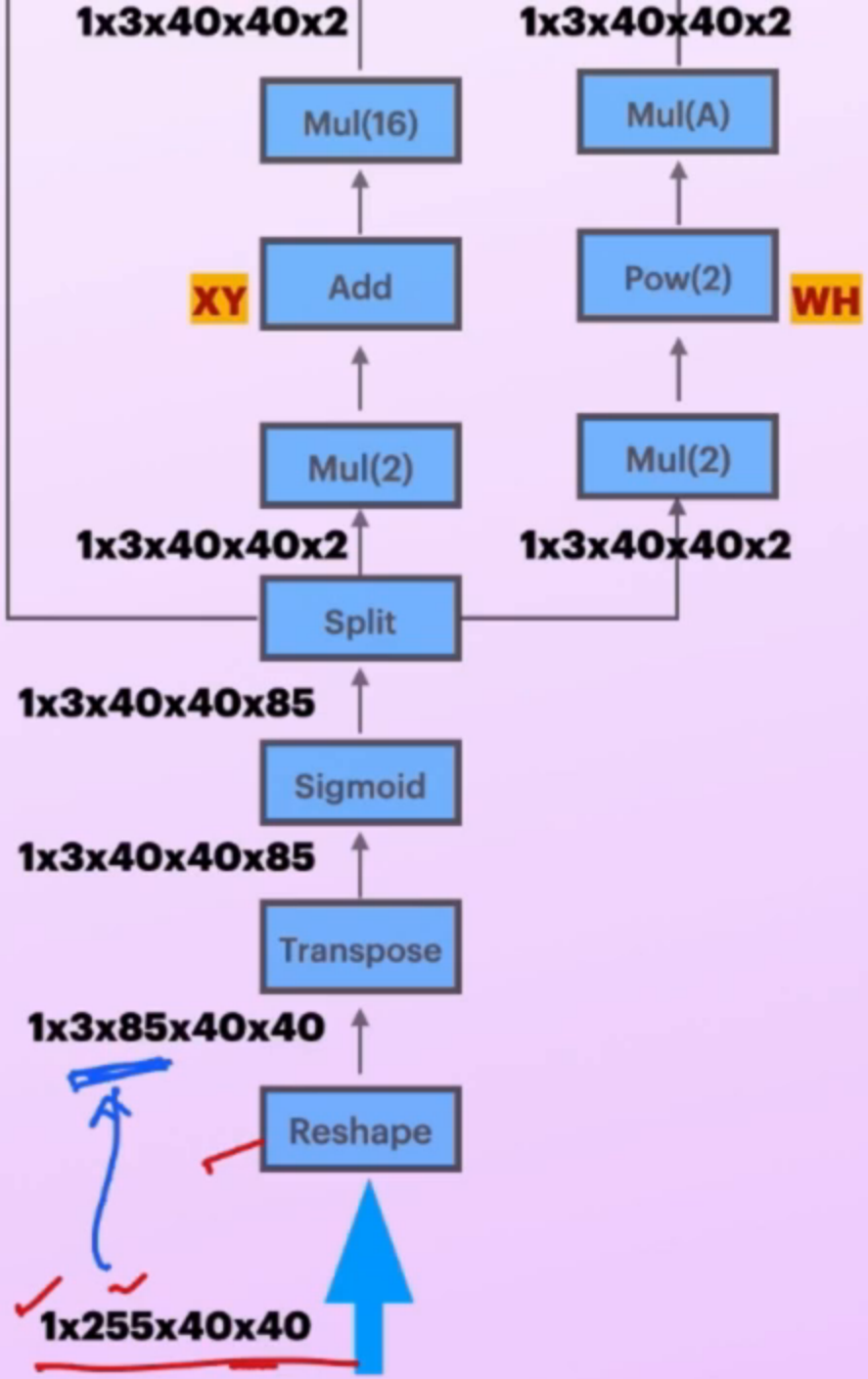

Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 ... ... p_C
3 anchor boxes per one grid cell
Box-Coordinates
Class Probabilities
80

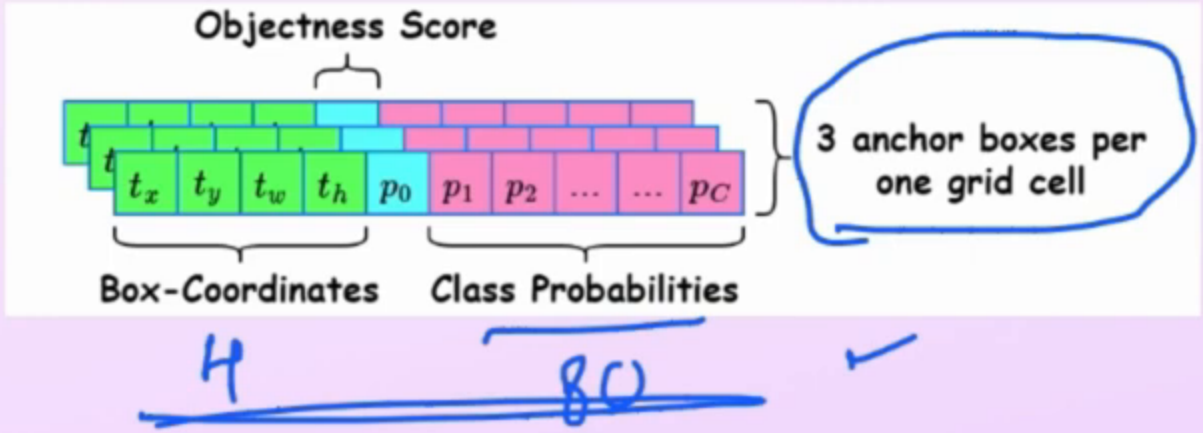

1x3x40x40x2
1x3x40x40x2
Mul(16)
Mul(A)
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
1x3x40x40x2
1x3x40x40x2
Split
1x3x40x40x85
Sigmoid
1x3x40x40x85
Transpose
1x3x85x40x40
Reshape
1x255x40x40

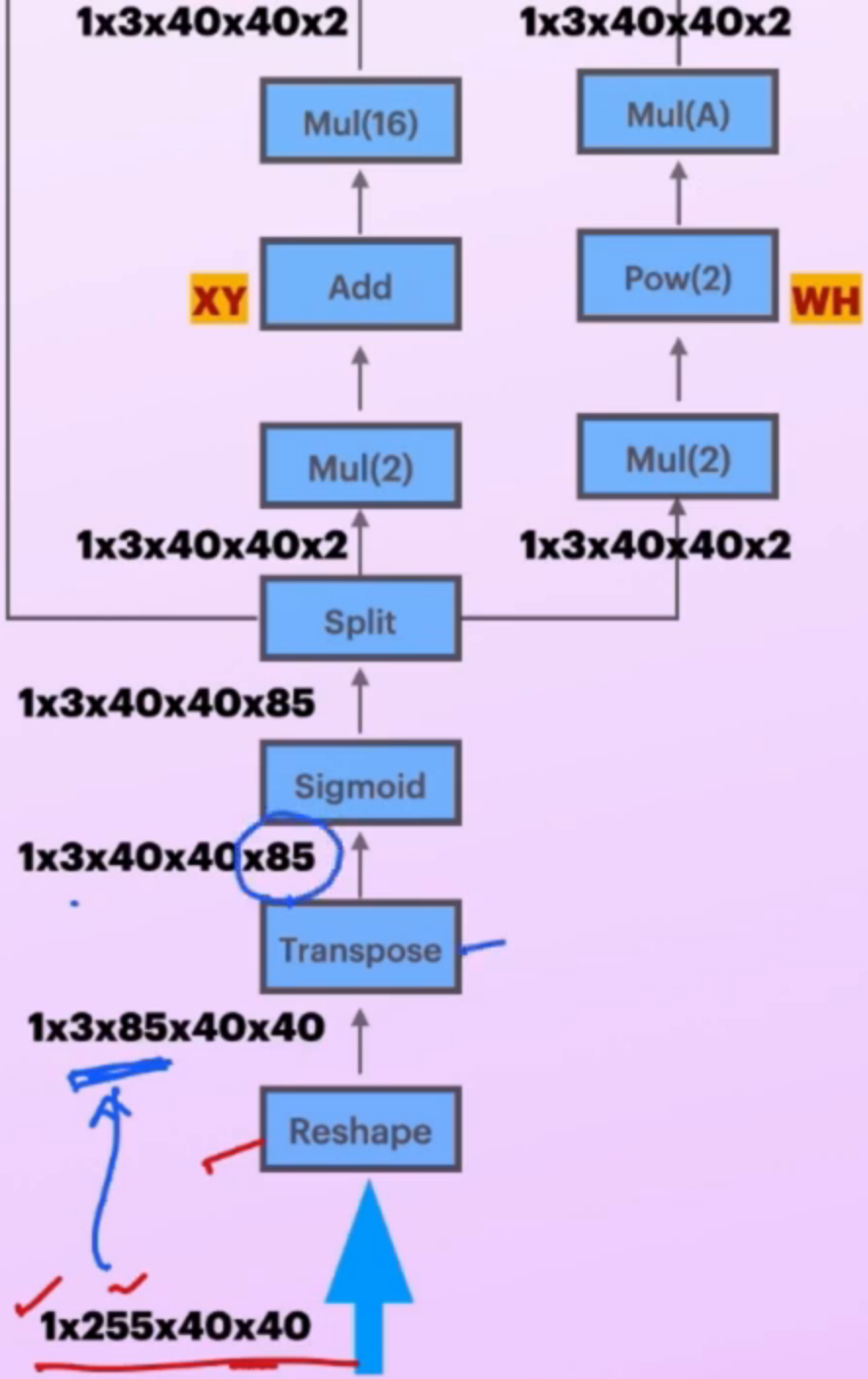

Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 ... ... p_C
3 anchor boxes per one grid cell
Box-Coordinates
Class Probabilities
4
80

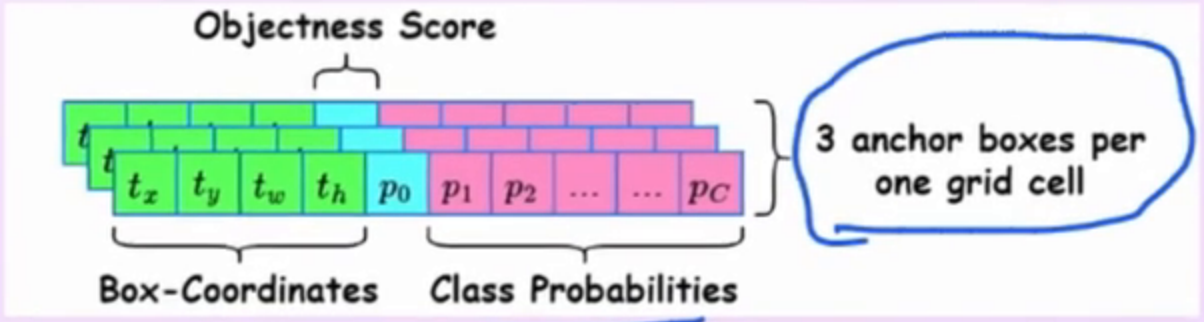

1x3x40x40x2
1x3x40x40x2
Mul(16)
Mul(A)
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
1x3x40x40x2
1x3x40x40x2
Split
1x3x40x40x85
Sigmoid
1x3x40x40x85
Transpose
1x3x85x40x40
Reshape
1x255x40x40

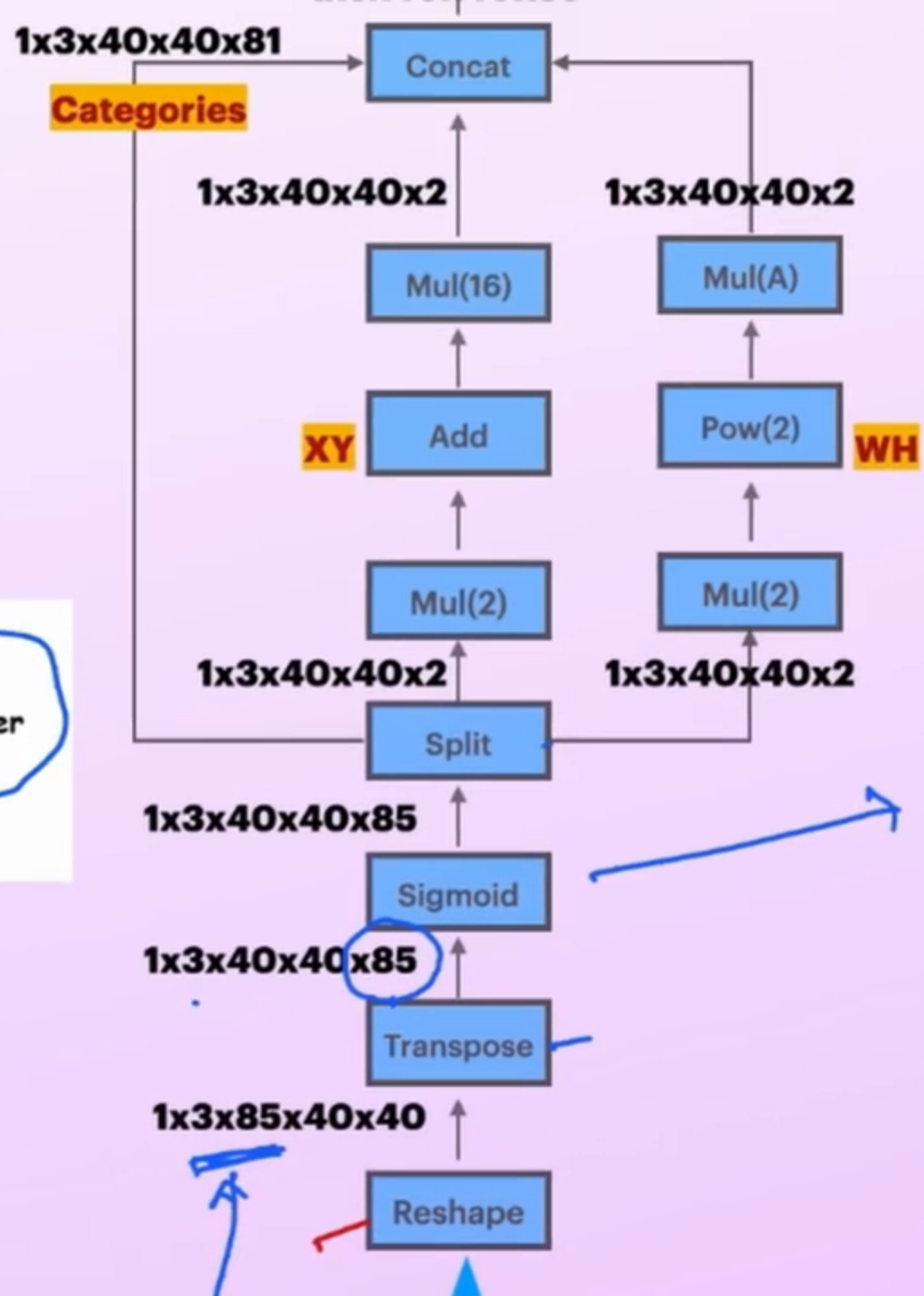

Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 ... ... p_C
3 anchor boxes per one grid cell
Box-Coordinates
Class Probabilities

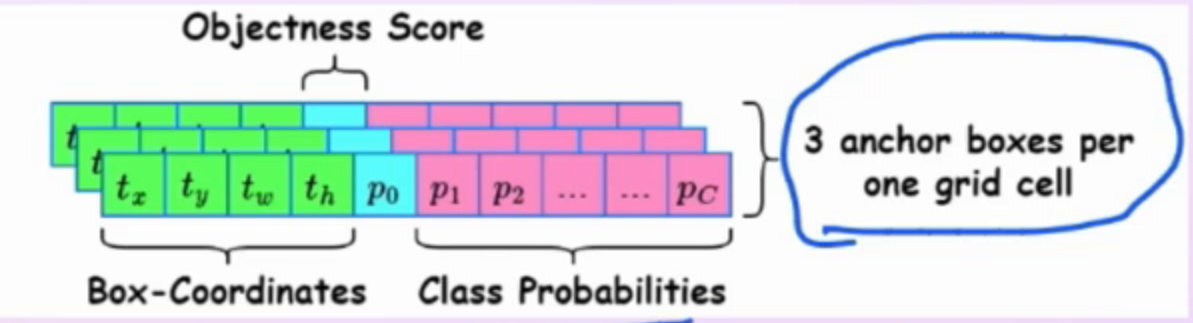

1x3x40x40x2
1x3x40x40x2
Mul(16)
Mul(A)
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
1x3x40x40x2
1x3x40x40x2
Split
1x3x40x40x85
Sigmoid
1x3x40x40x85
Transpose
1x3x85x40x40
Reshape
1x255x40x40

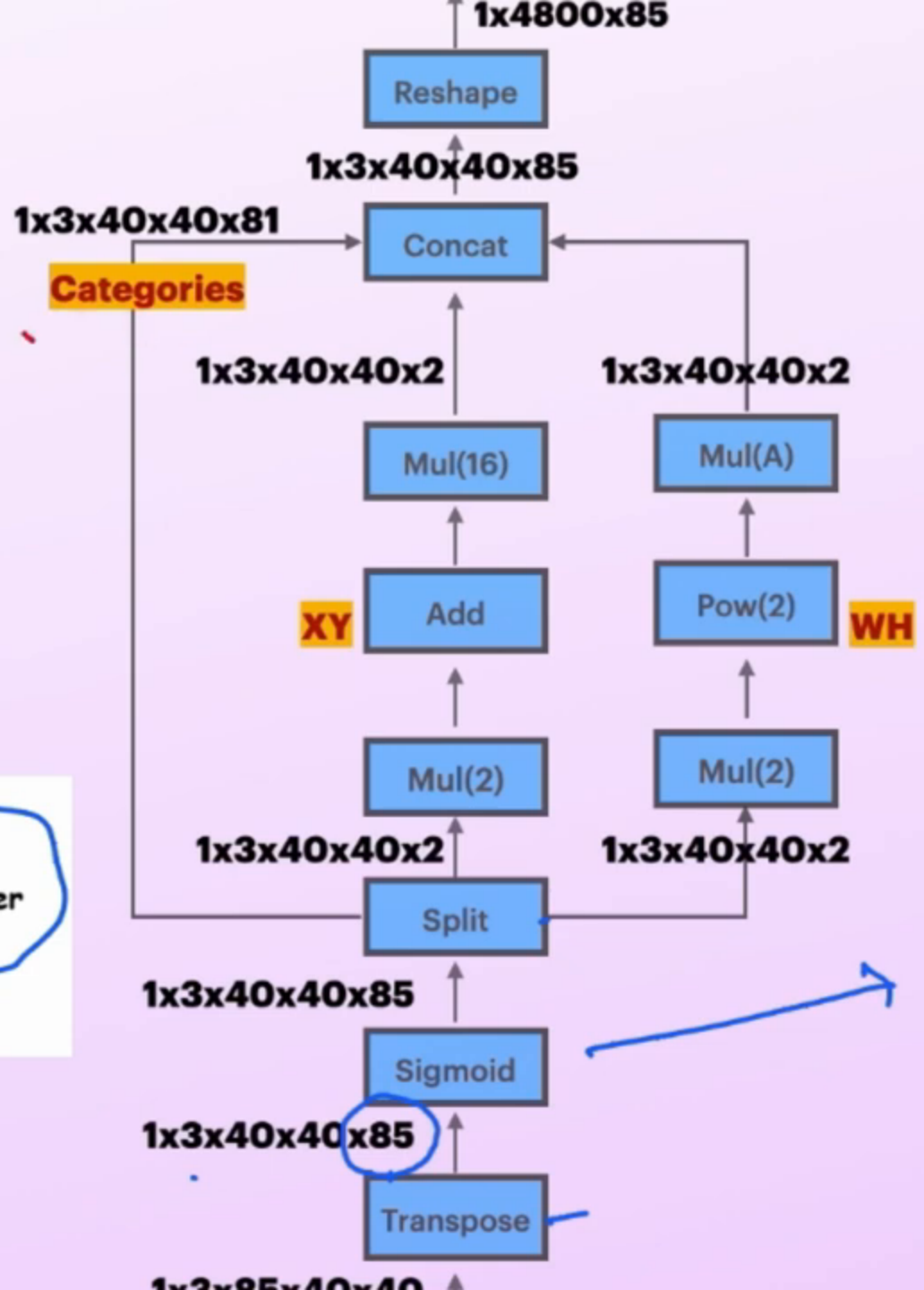

Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 ... ... p_C
3 anchor boxes per one grid cell
Box-Coordinates
Class Probabilities

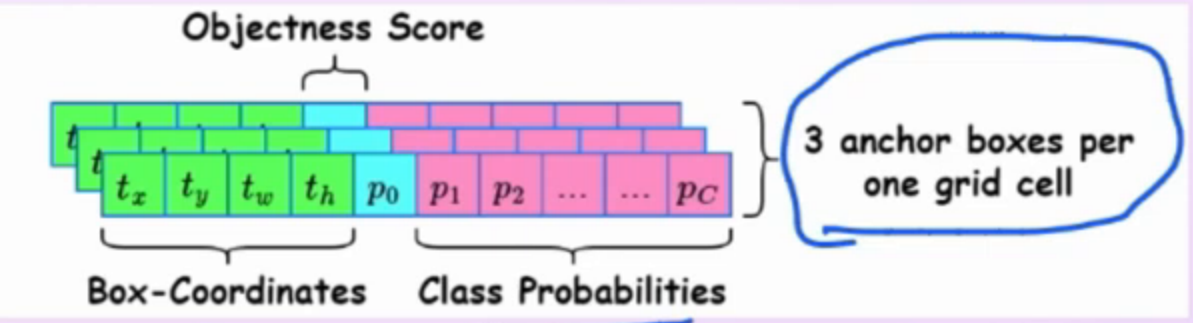

1x3x40x40x2
1x3x40x40x2
Mul(16)
Mul(A)
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
1x3x40x40x2
1x3x40x40x2
Split
1x3x40x40x85
Sigmoid
1x3x40x40x85
Transpose
1x3x85x40x40
Reshape
1x255x40x40

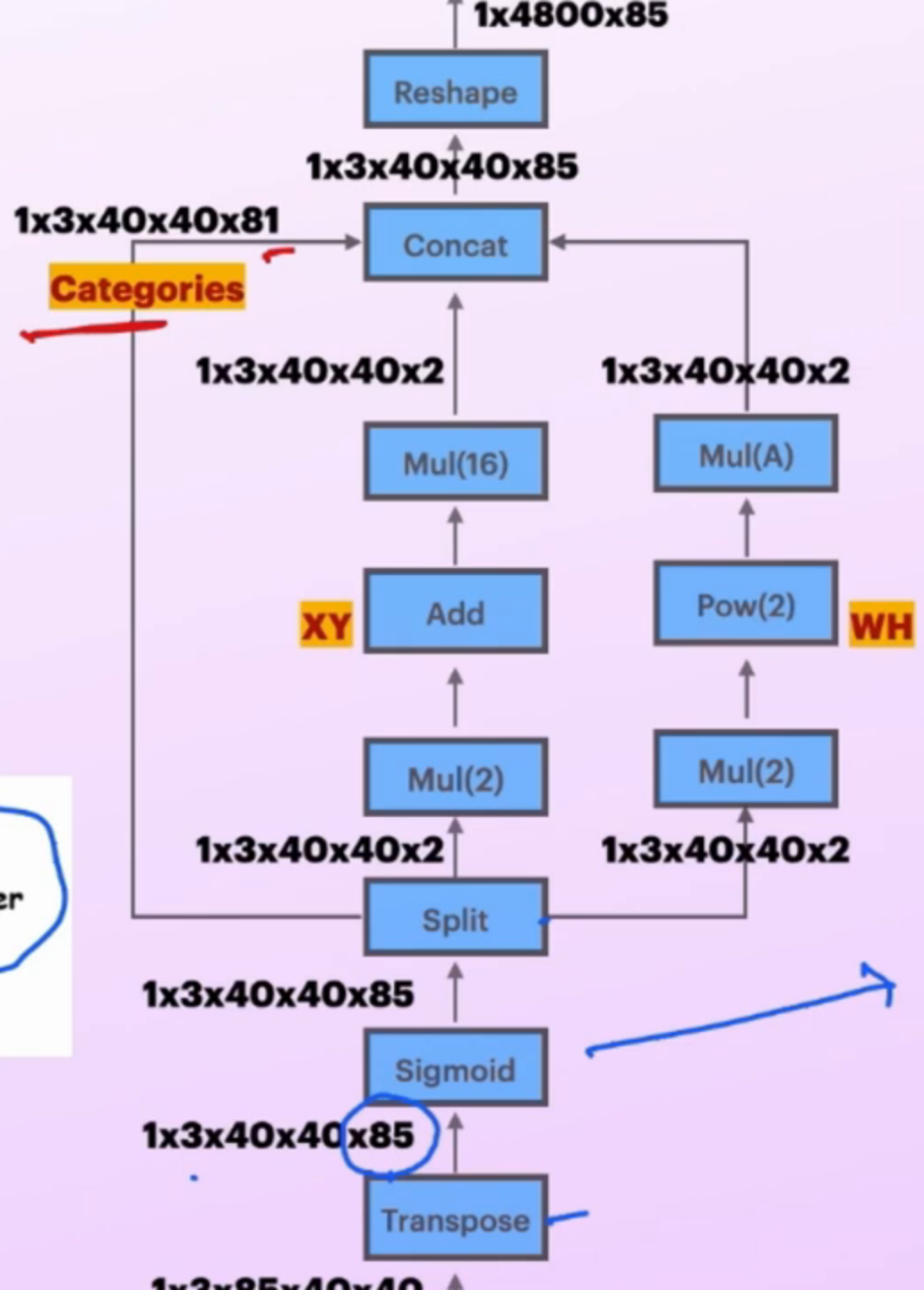

Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 … … p_C
3 anchor boxes per one grid cell
Box-Coordinates
Class Probabilities

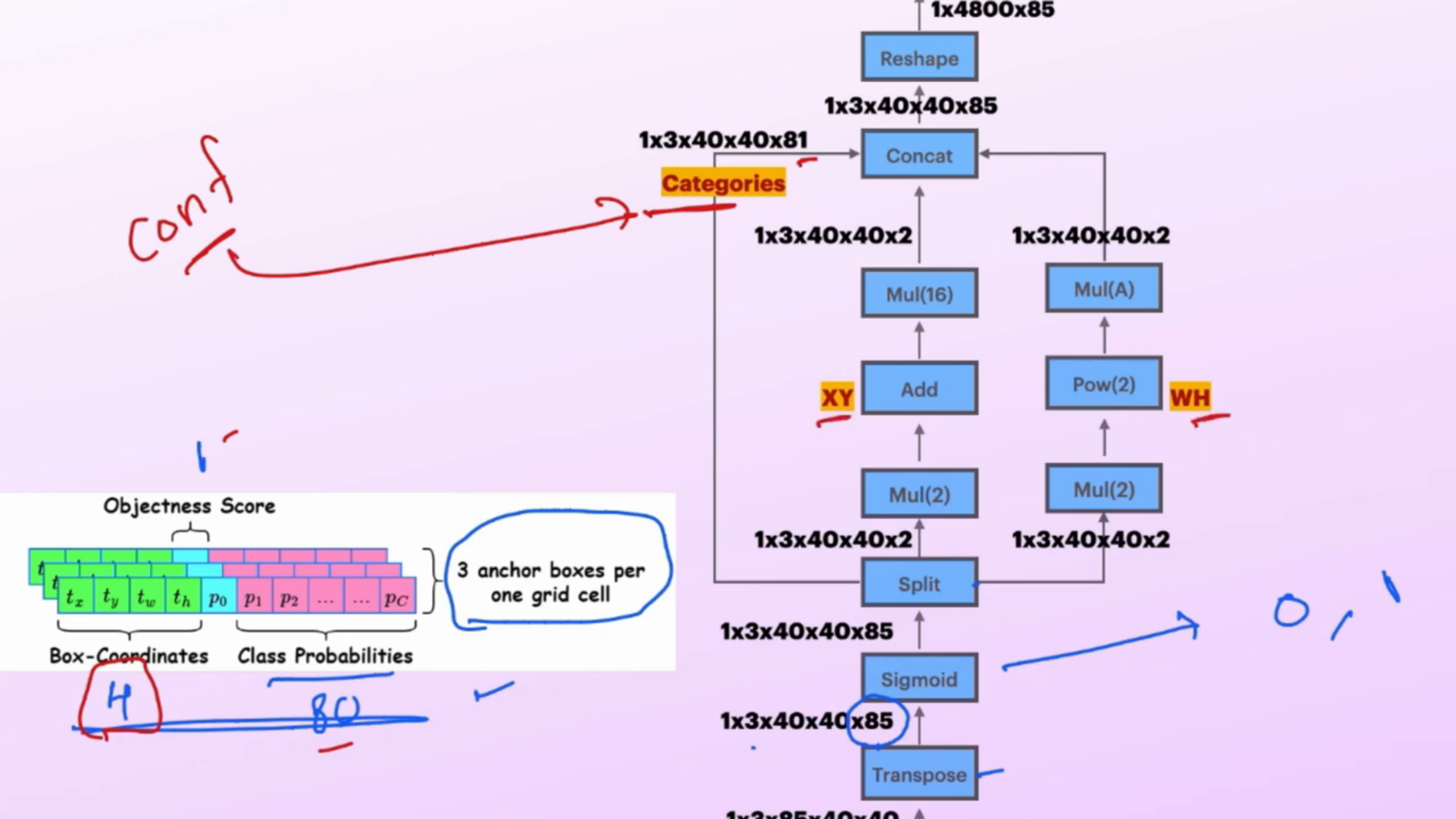

1x3x40x40x81
Concat
Categories
1x3x40x40x2
1x3x40x40x2
Mul(16)
Mul(A)
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
1x3x40x40x2
1x3x40x40x2
Split
1x3x40x40x85
Sigmoid
1x3x40x40x85
Transpose
1x3x85x40x40
Reshape

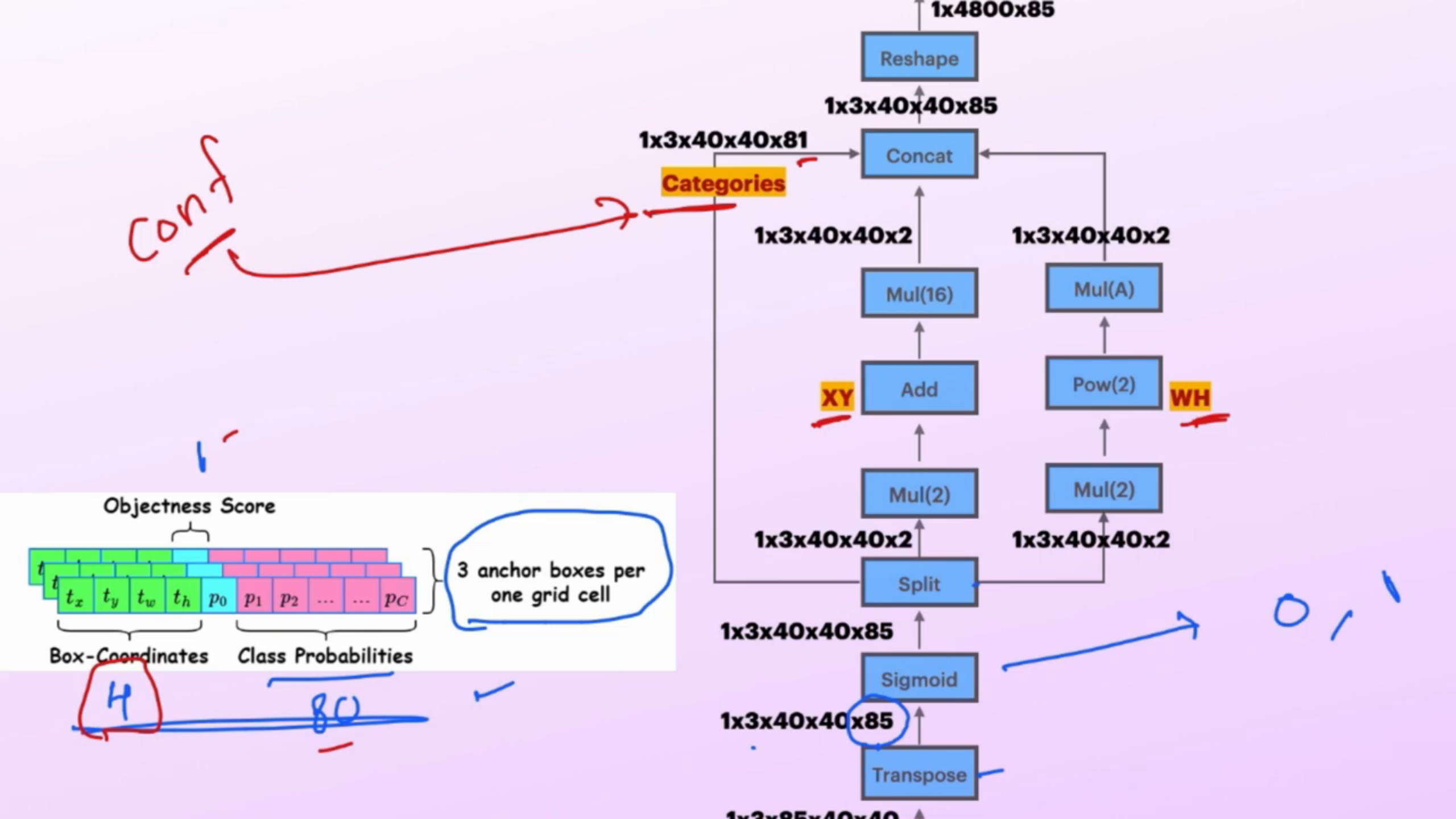

Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 ... ... p_C
3 anchor boxes per one grid cell
Box-Coordinates
Class Probabilities

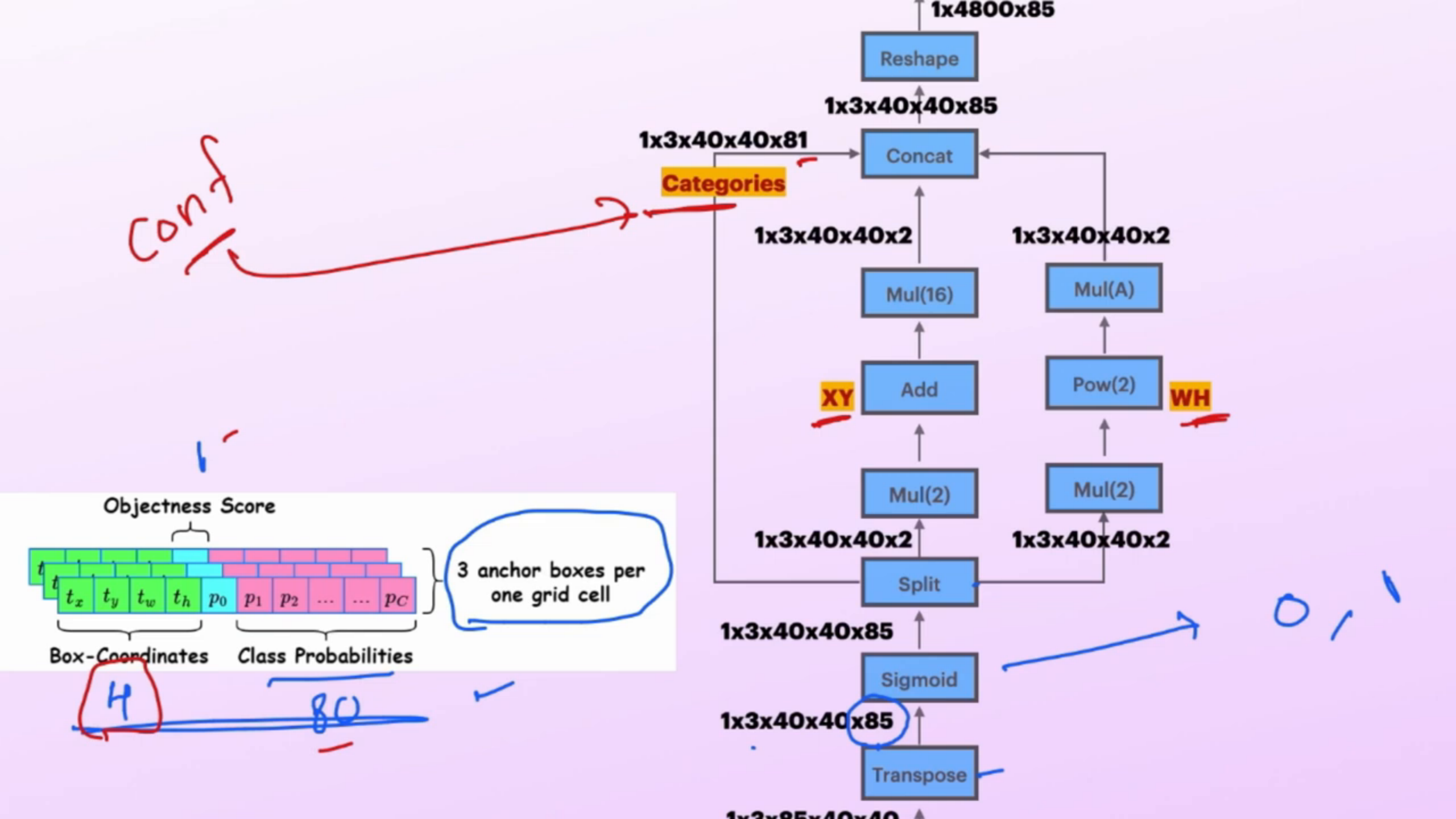

1x4800x85
Reshape
1x3x40x40x85
1x3x40x40x81
Concat
Categories
1x3x40x40x2
1x3x40x40x2
Mul(16)
Mul(A)
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
1x3x40x40x2
1x3x40x40x2
Split
1x3x40x40x85
Sigmoid
1x3x40x40x85
Transpose


Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 ... ... p_C
3 anchor boxes per one grid cell
Box-Coordinates
Class Probabilities

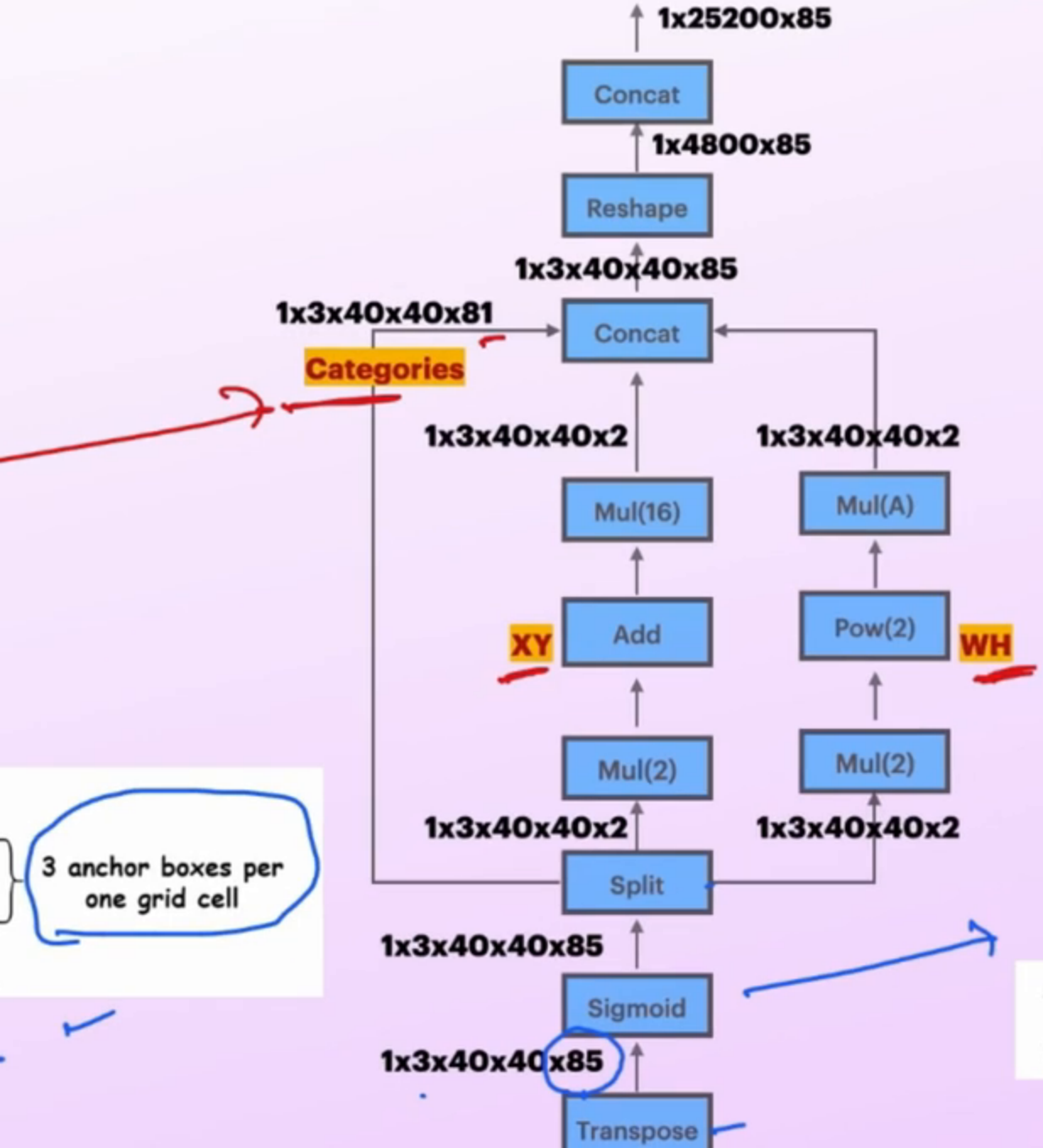

1x4800x85
Reshape
1x3x40x40x85
1x3x40x40x81
Concat
Categories
1x3x40x40x2
1x3x40x40x2
Mul(16)
Mul(A)
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
1x3x40x40x2
1x3x40x40x2
Split
1x3x40x40x85
Sigmoid
1x3x40x40x85
Transpose

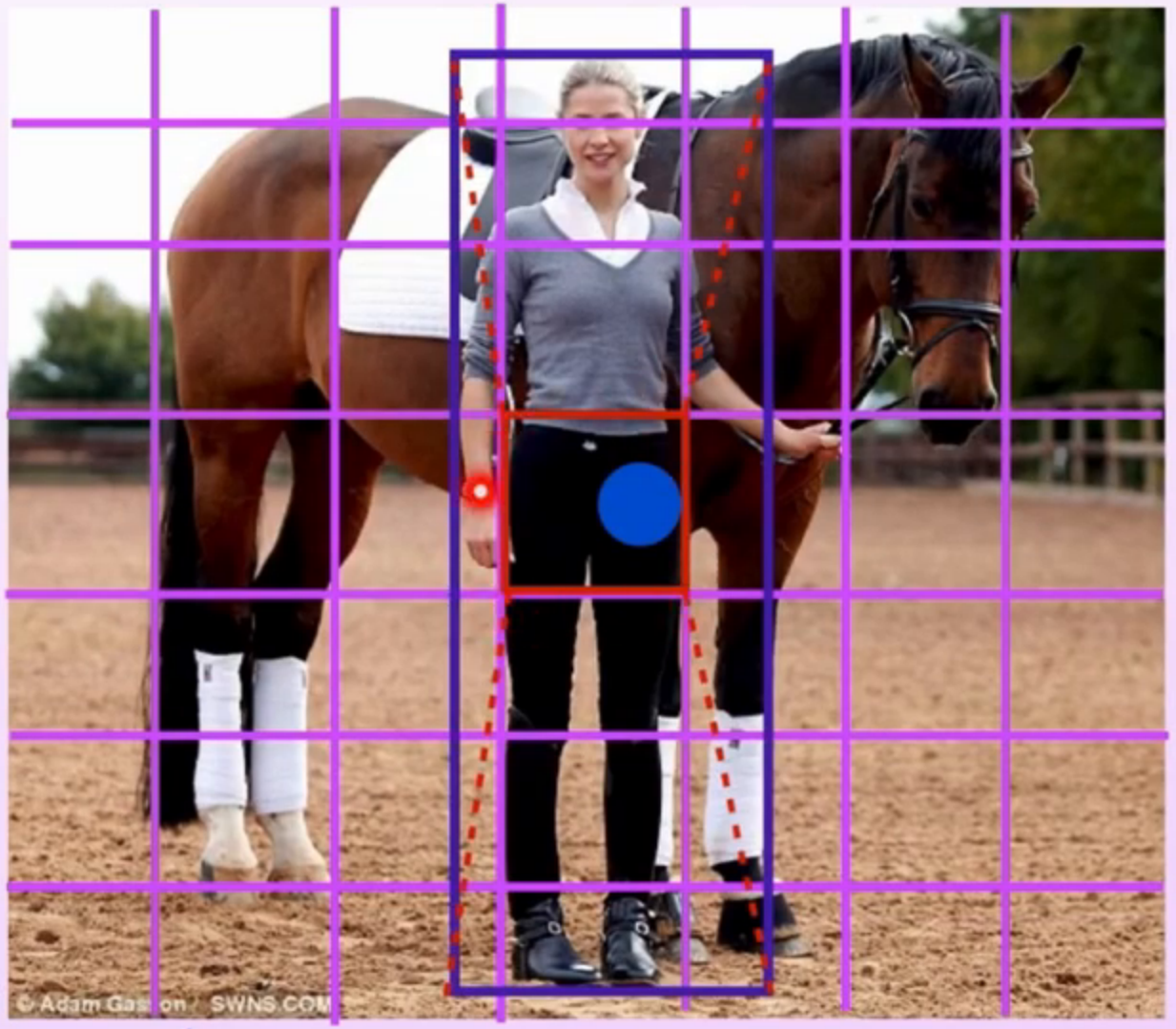

1x4800x85
Reshape
1x3x40x40x85
1x3x40x40x81
Concat
Categories
1x3x40x40x2
1x3x40x40x2
Mul(16)
Mul(A)
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
1x3x40x40x2
1x3x40x40x2
Split
1x3x40x40x85
Sigmoid
1x3x40x40x85
Transpose
Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 ... ... p_C
3 anchor boxes per one grid cell
Box-Coordinates
Class Probabilities
4
80


1x4800x85
Reshape
1x3x40x40x85
1x3x40x40x81
Concat
Categories
1x3x40x40x2
1x3x40x40x2
Mul(16)
Mul(A)
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
1x3x40x40x2
1x3x40x40x2
Split
1x3x40x40x85
Sigmoid
1x3x40x40x85
Transpose
Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 ... ... p_C
3 anchor boxes per one grid cell
Box-Coordinates
Class Probabilities
4
80


1x4800x85
Reshape
1x3x40x40x85
1x3x40x40x81
Concat
Categories
1x3x40x40x2
1x3x40x40x2
Mul(16)
Mul(A)
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
1x3x40x40x2
1x3x40x40x2
Split
1x3x40x40x85
Sigmoid
1x3x40x40x85
Transpose
Objectness Score
t_x t_y t_w t_h p_0 p_1 p_2 ... ... p_C
3 anchor boxes per one grid cell
Box-Coordinates
Class Probabilities

NMS & Postprocessing





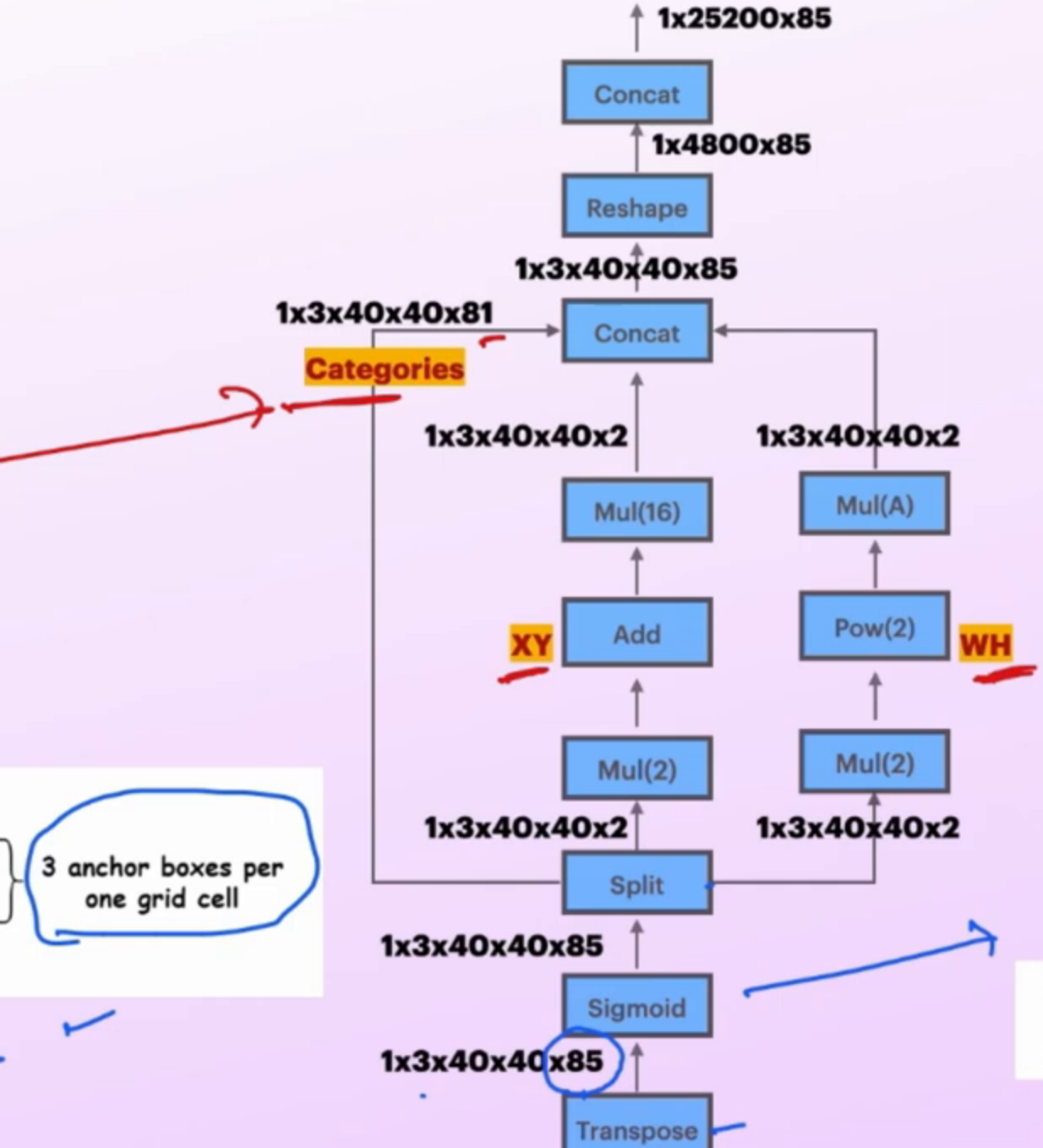
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$

$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

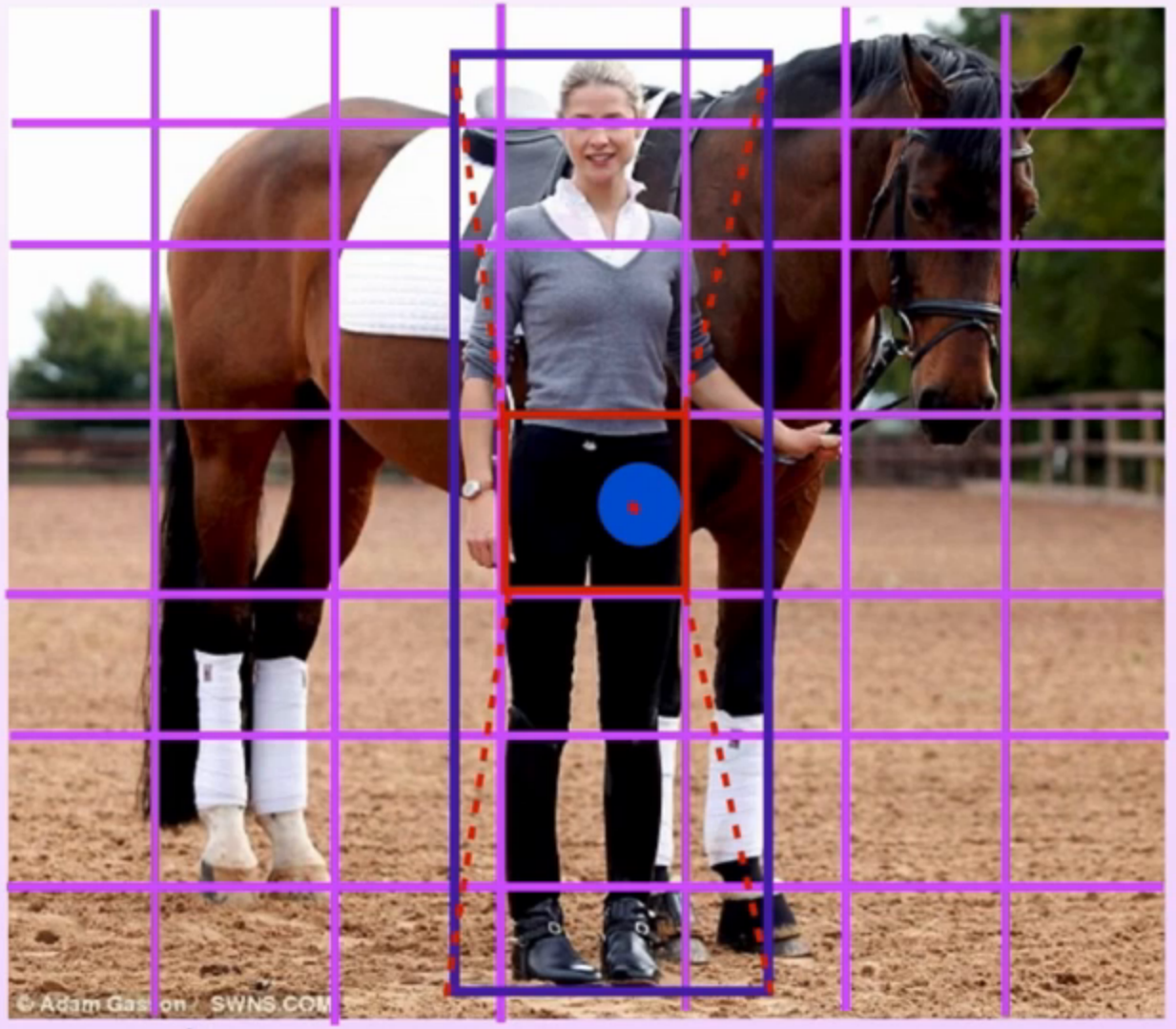
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$

$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

NMS & Postprocessing



3 anchor boxes per one grid cell



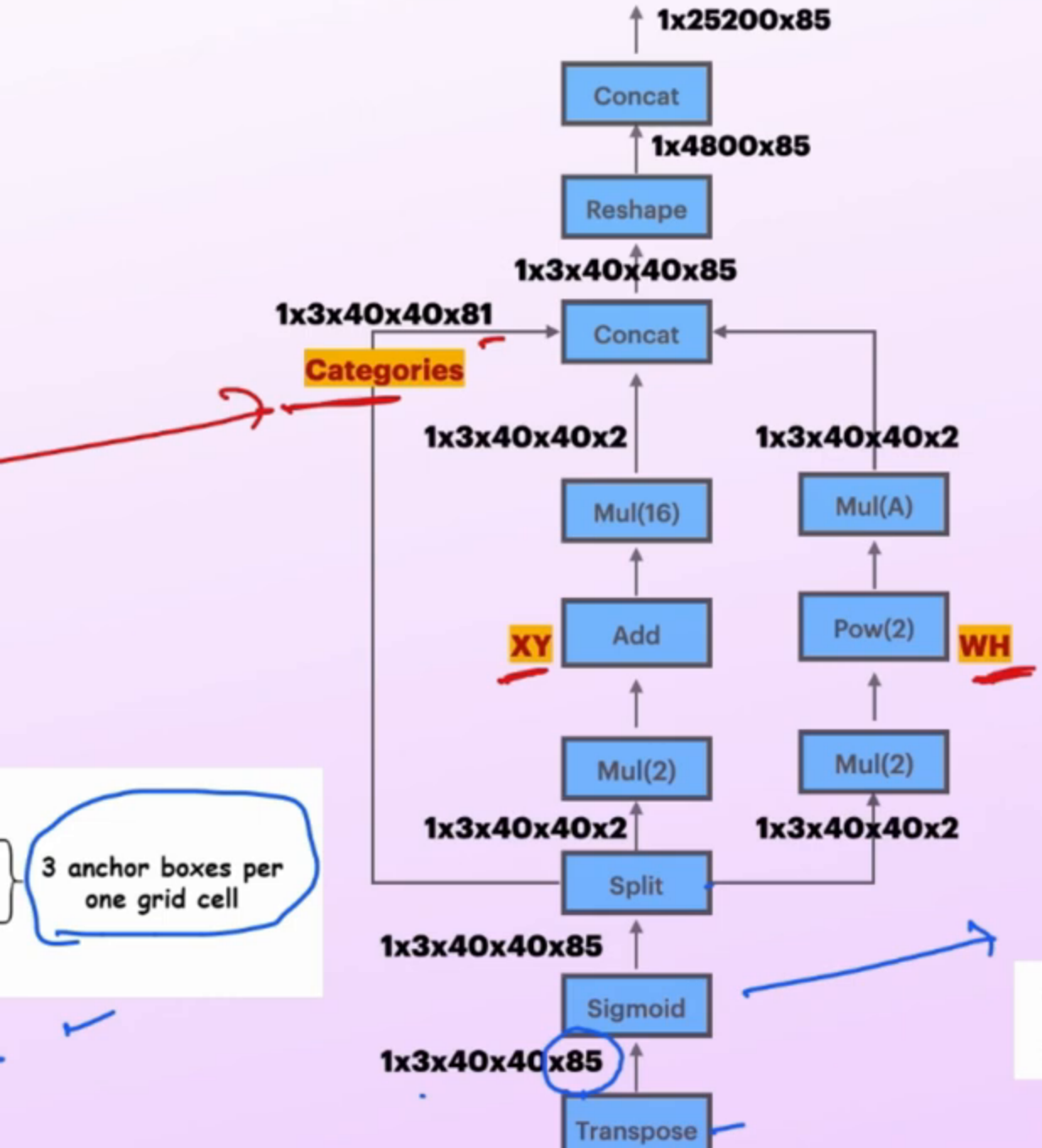
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$

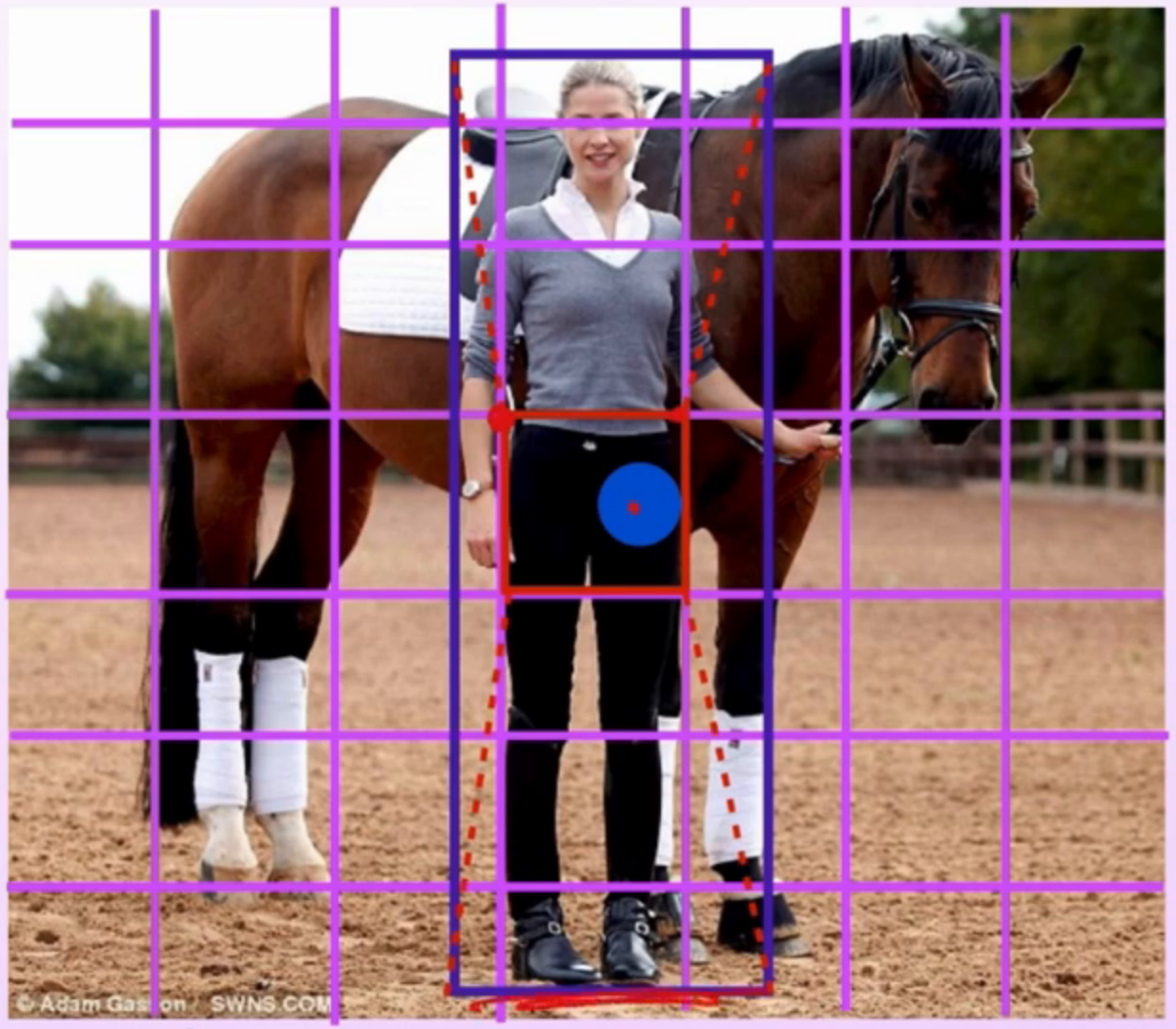
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

$$w = p_w \times (2 \cdot \sigma(t_w))^2$$

$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

NMS & Postprocessing





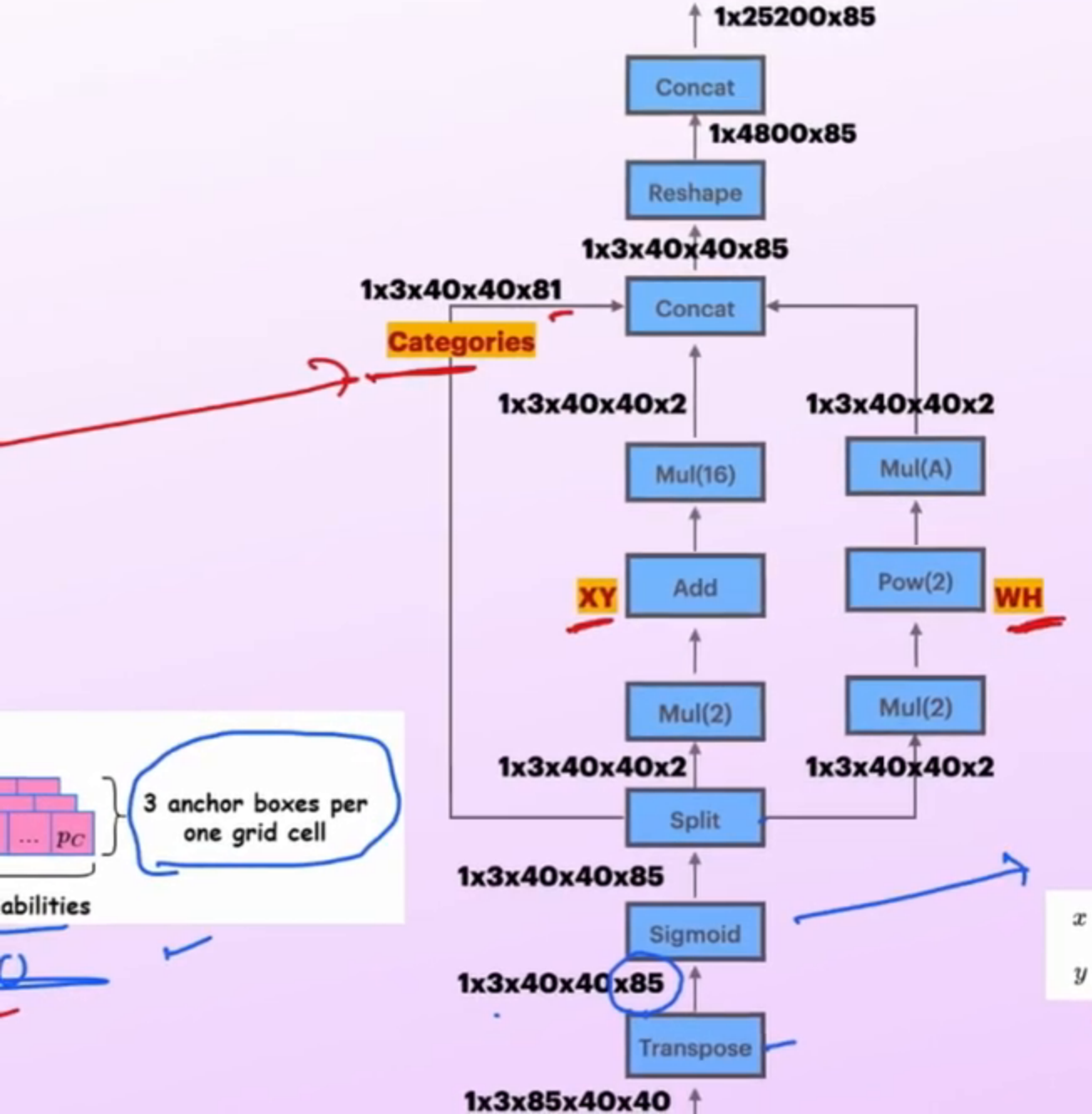
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$

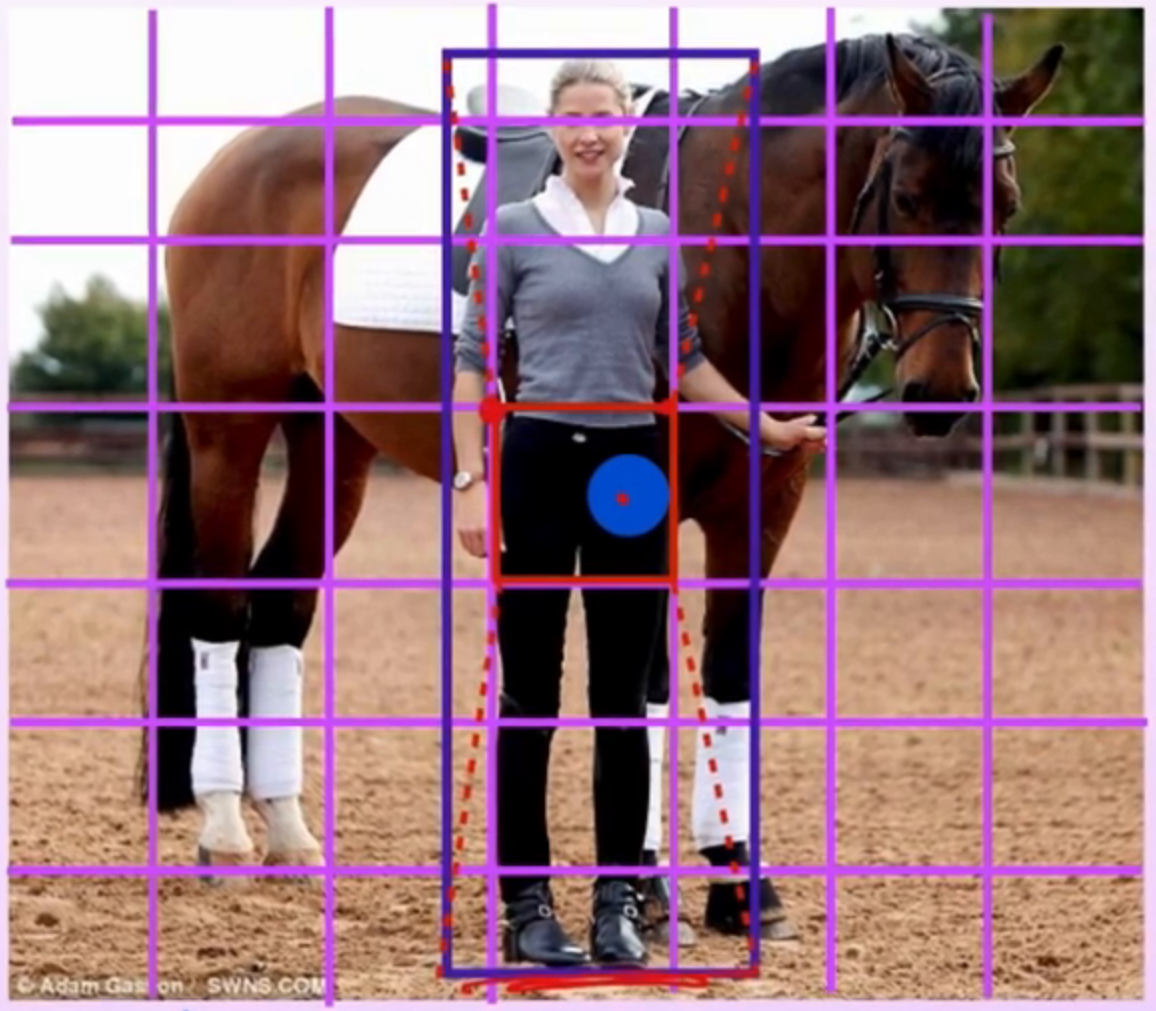
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

$$w = p_w \times (2 \cdot \sigma(t_w))^2$$

$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

NMS & Postprocessing

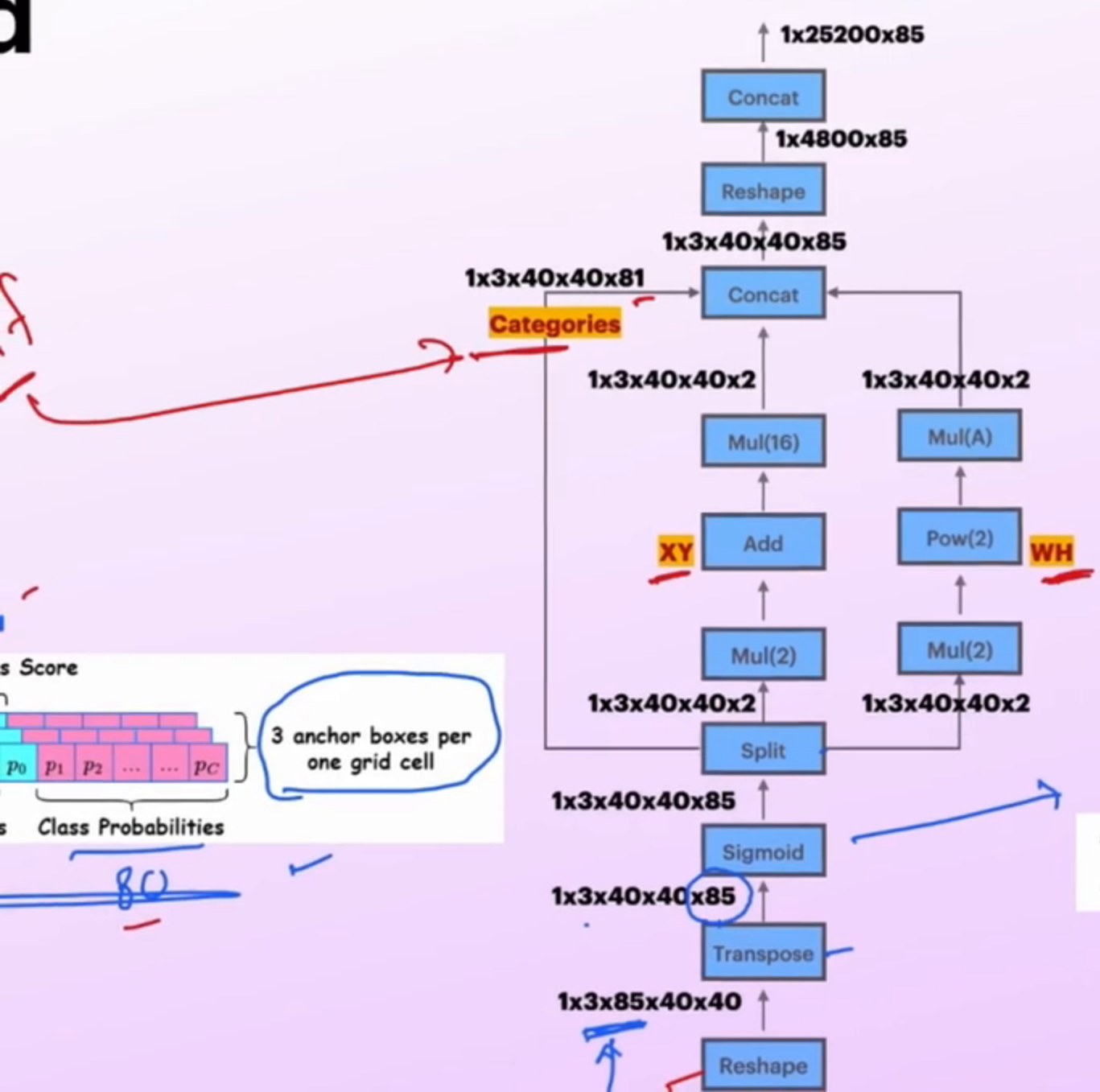


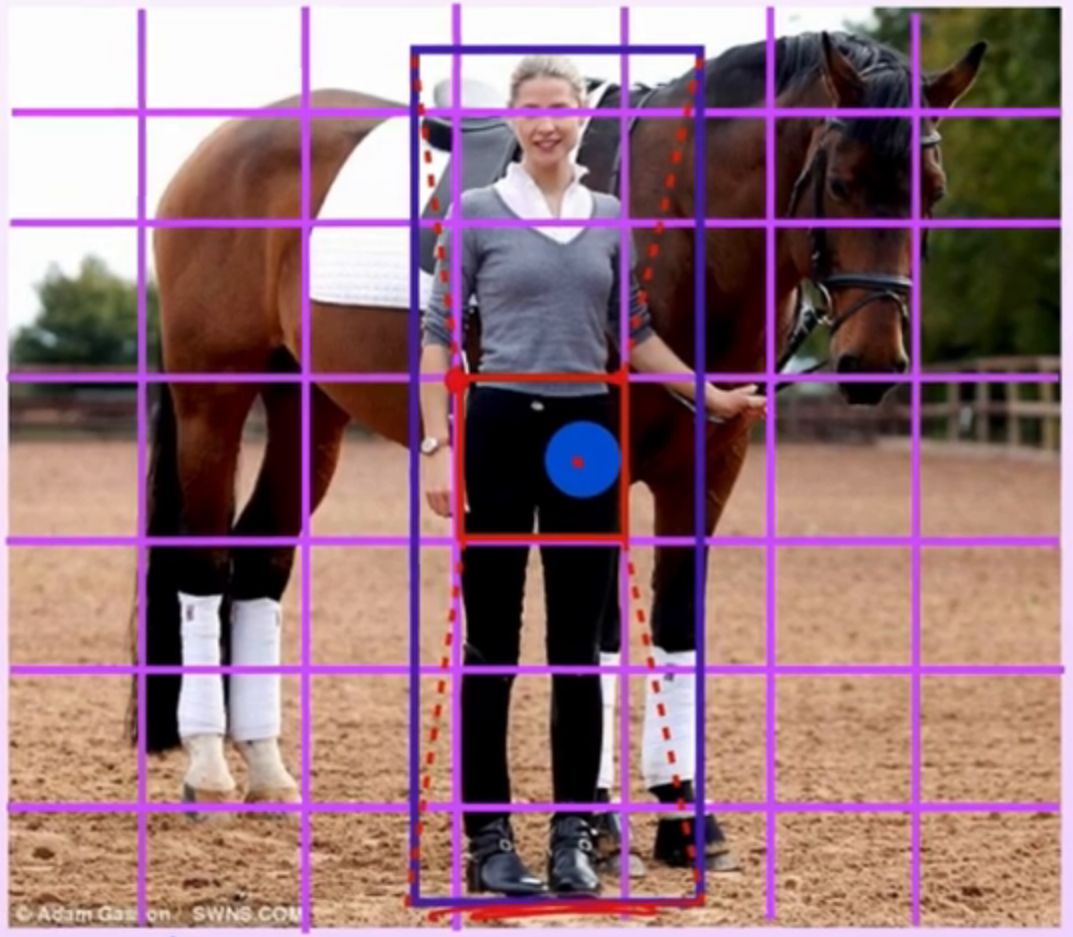


$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

NMS & Postprocessing



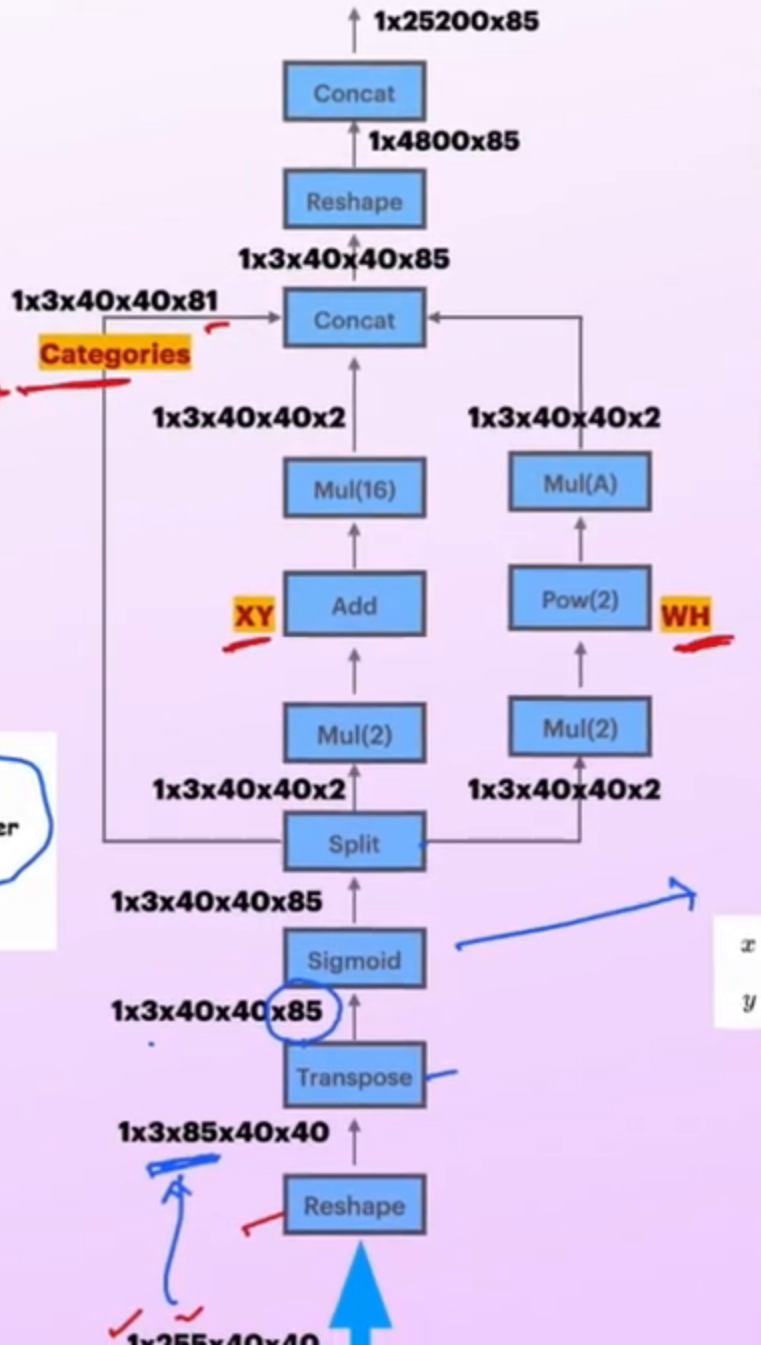


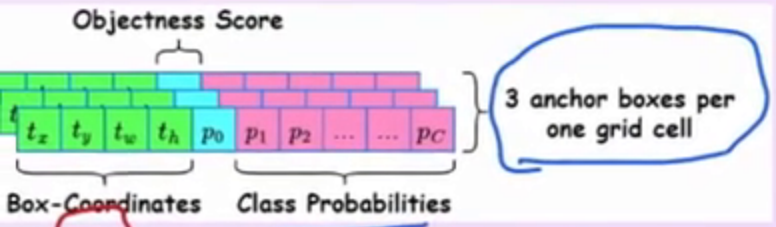
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

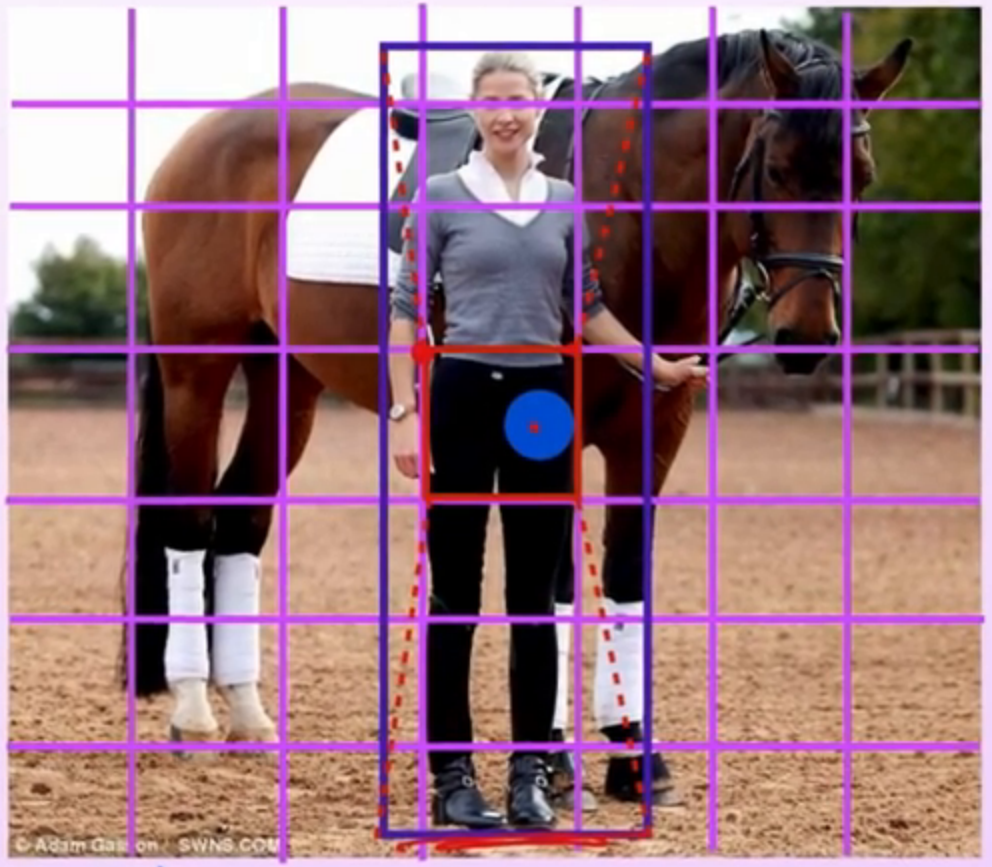
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





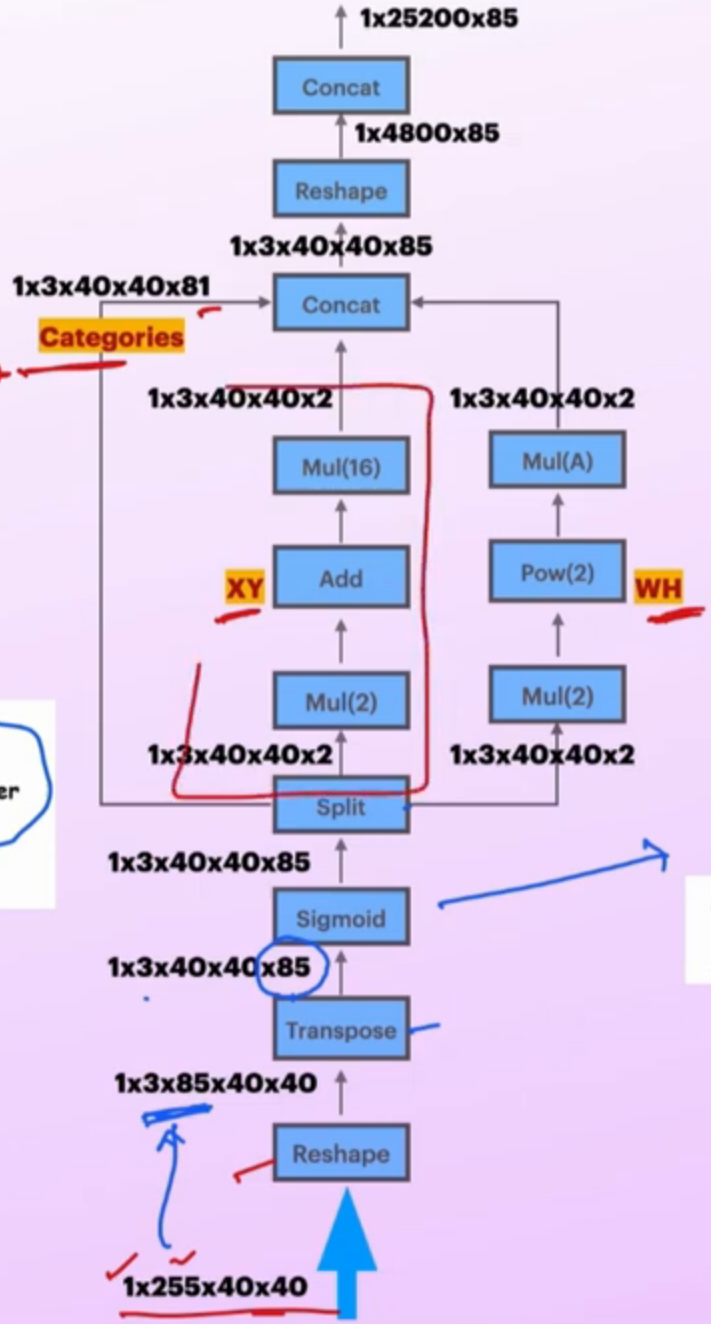


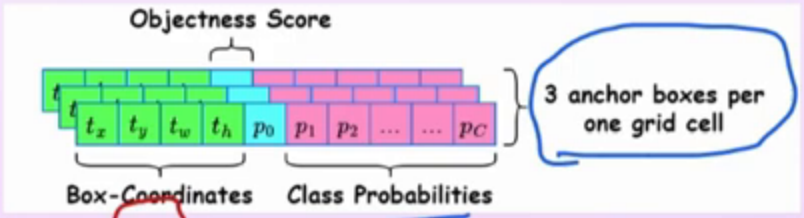
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

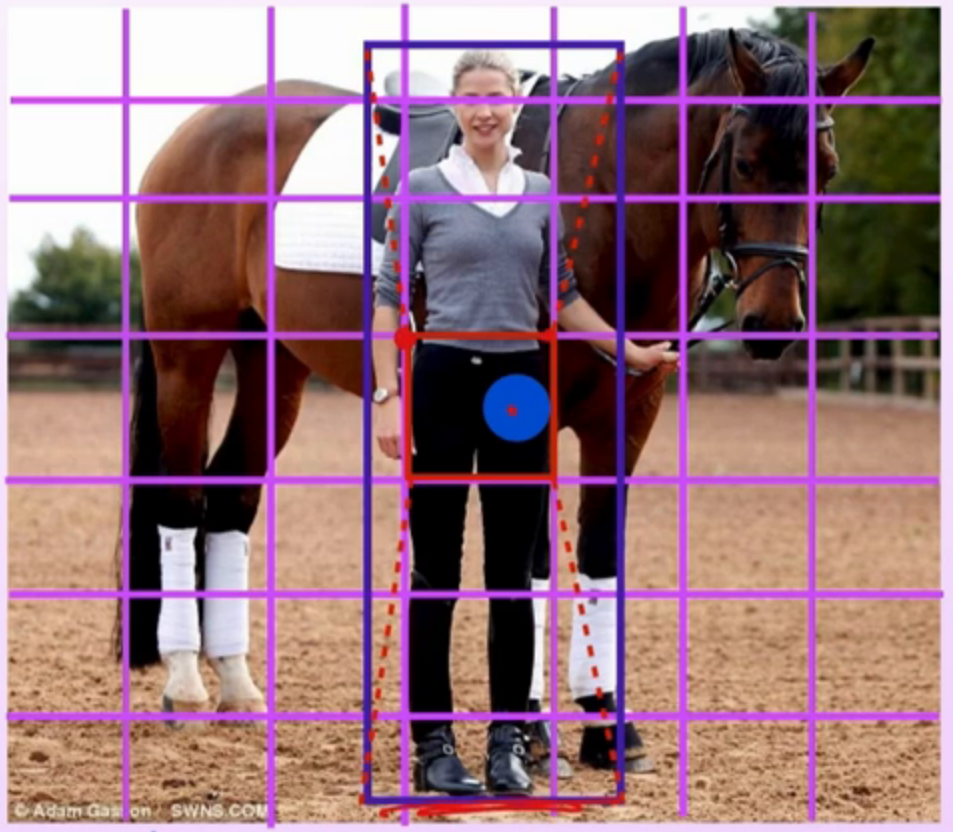
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





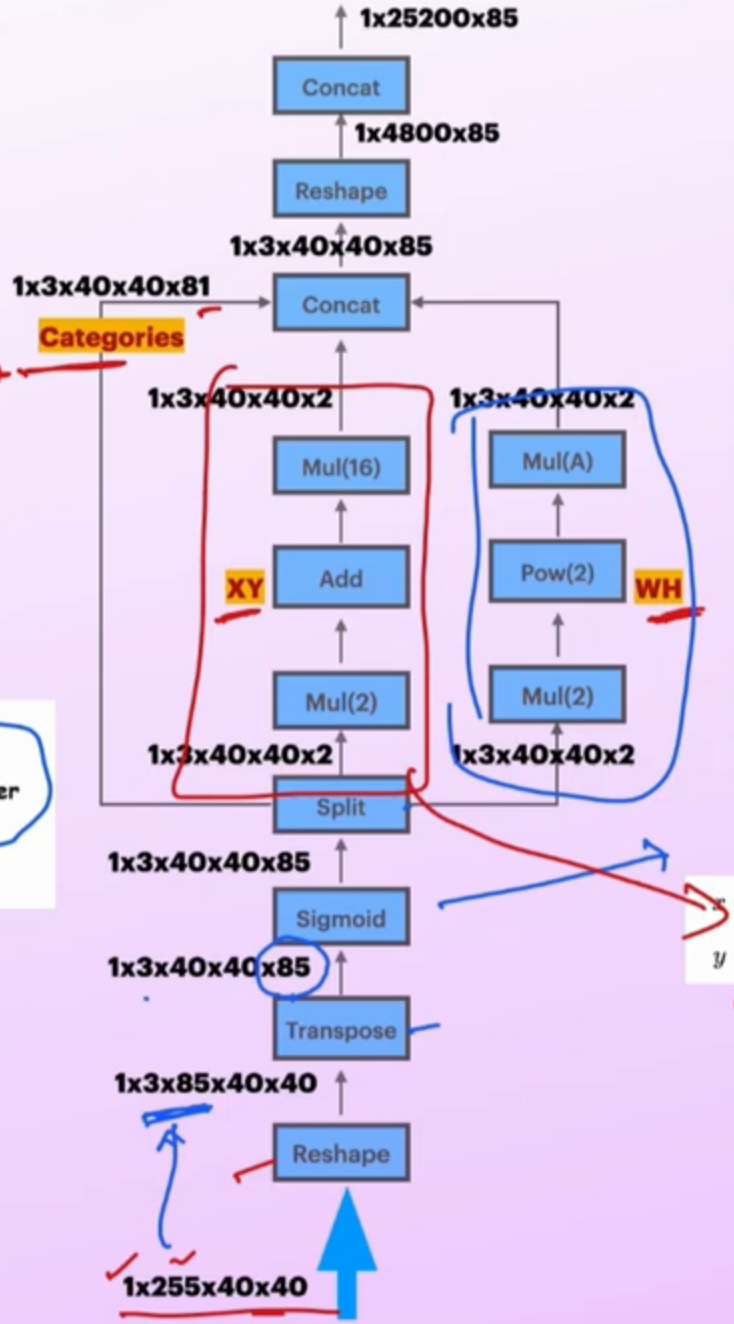


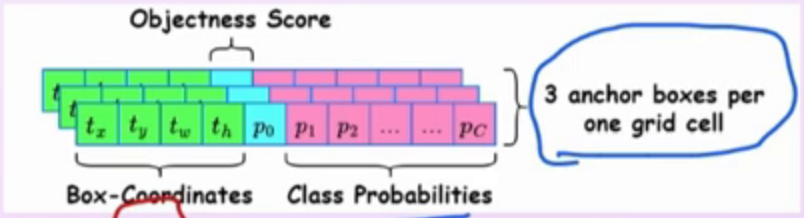
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

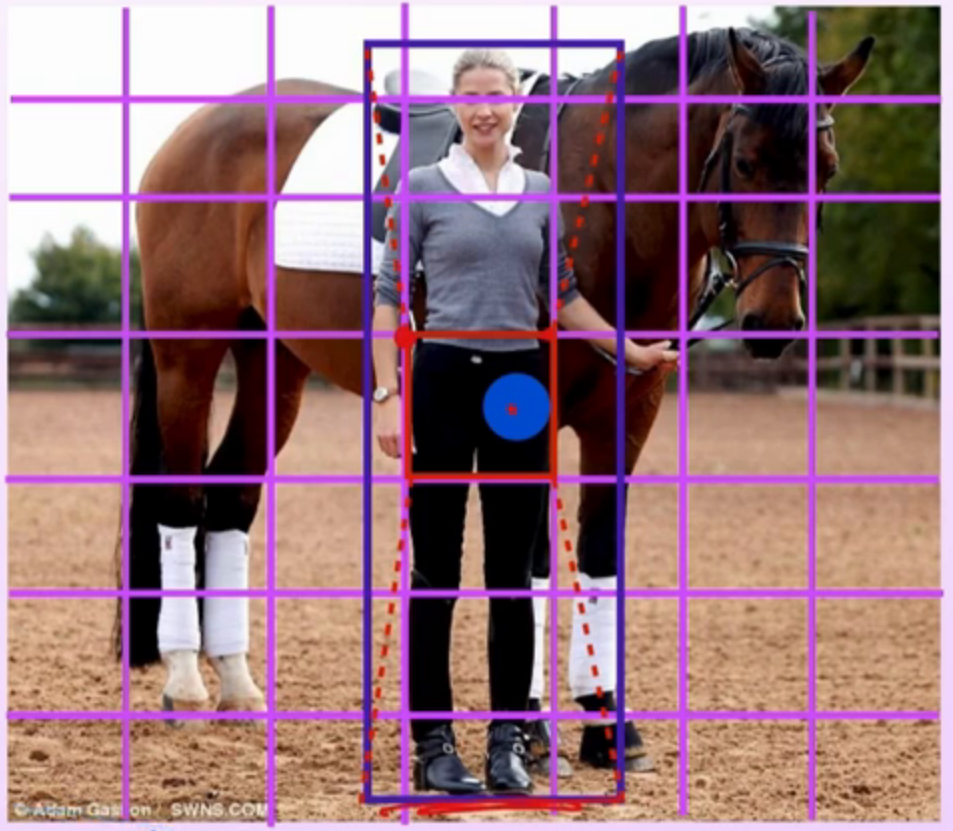
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





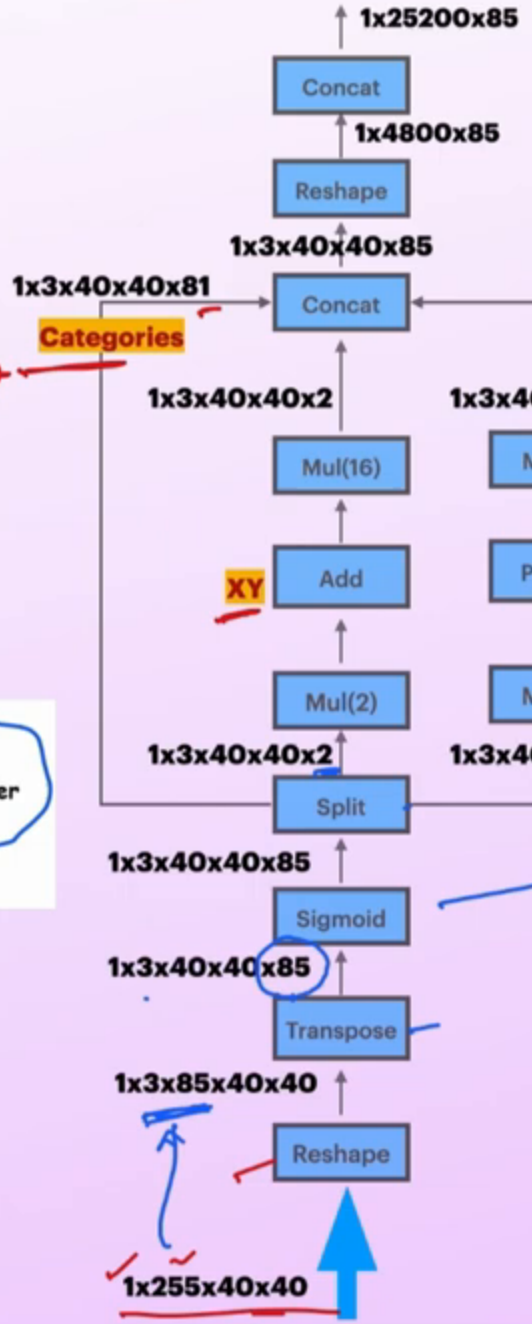


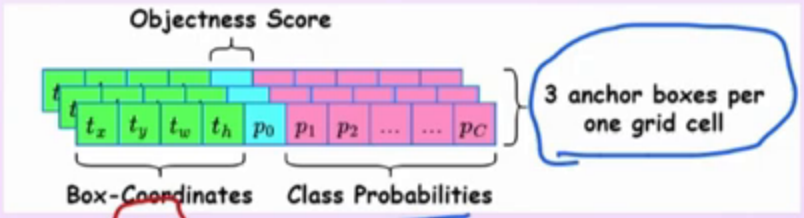
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

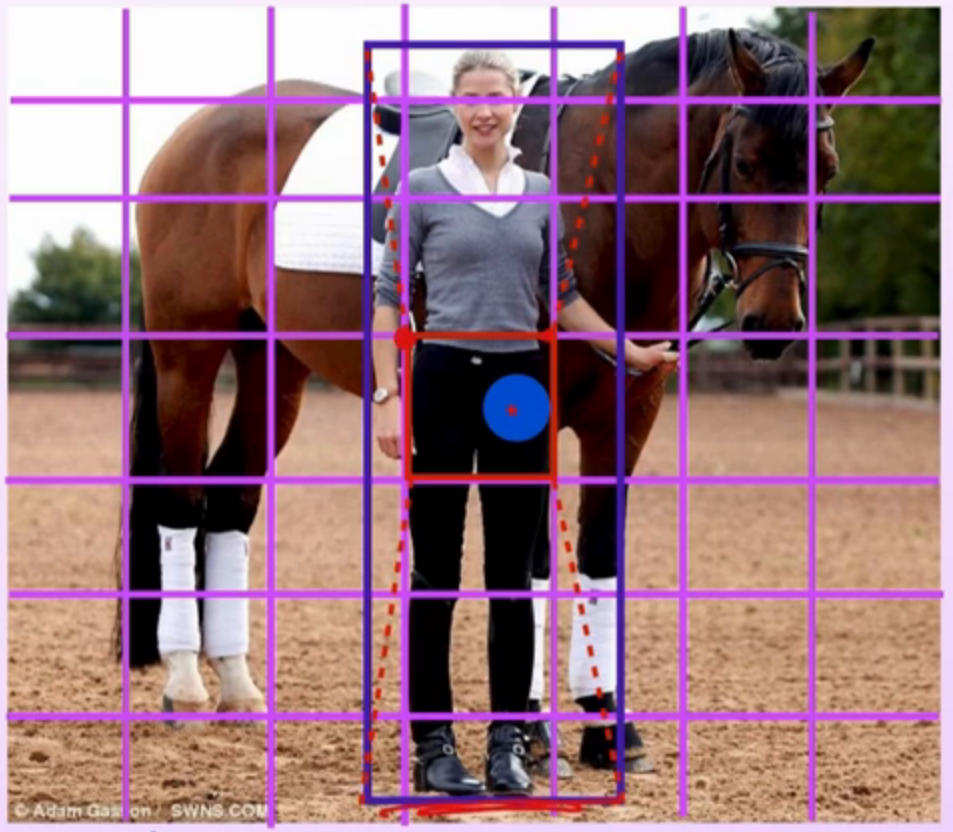
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





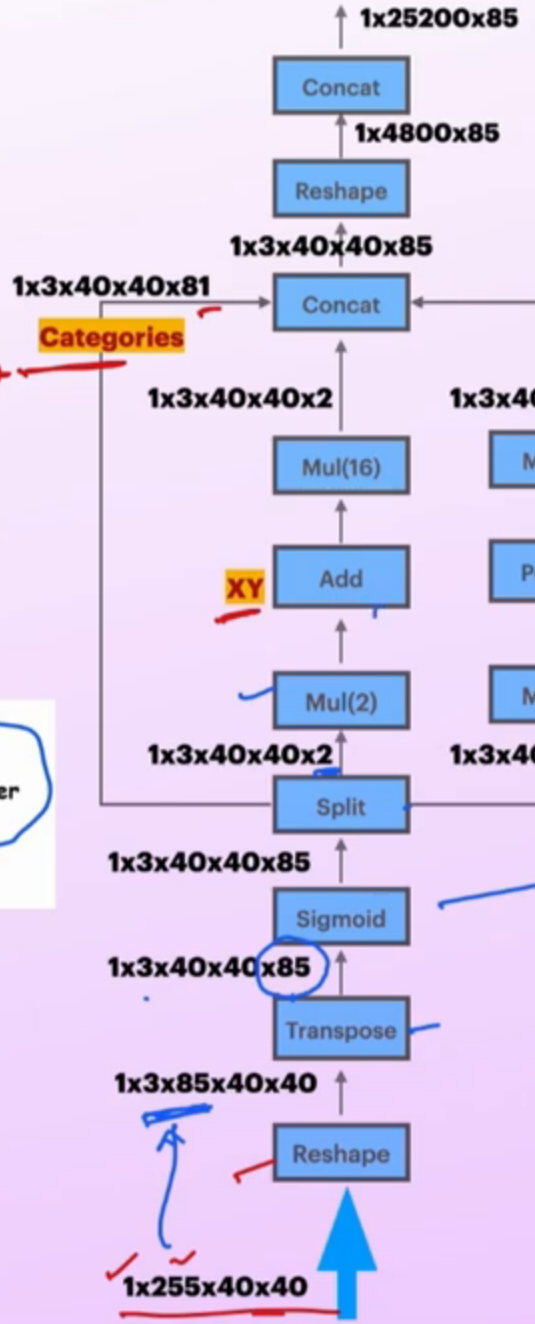


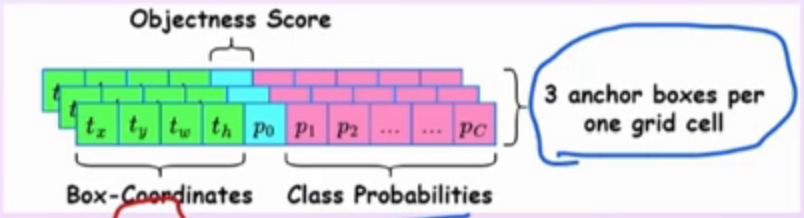
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

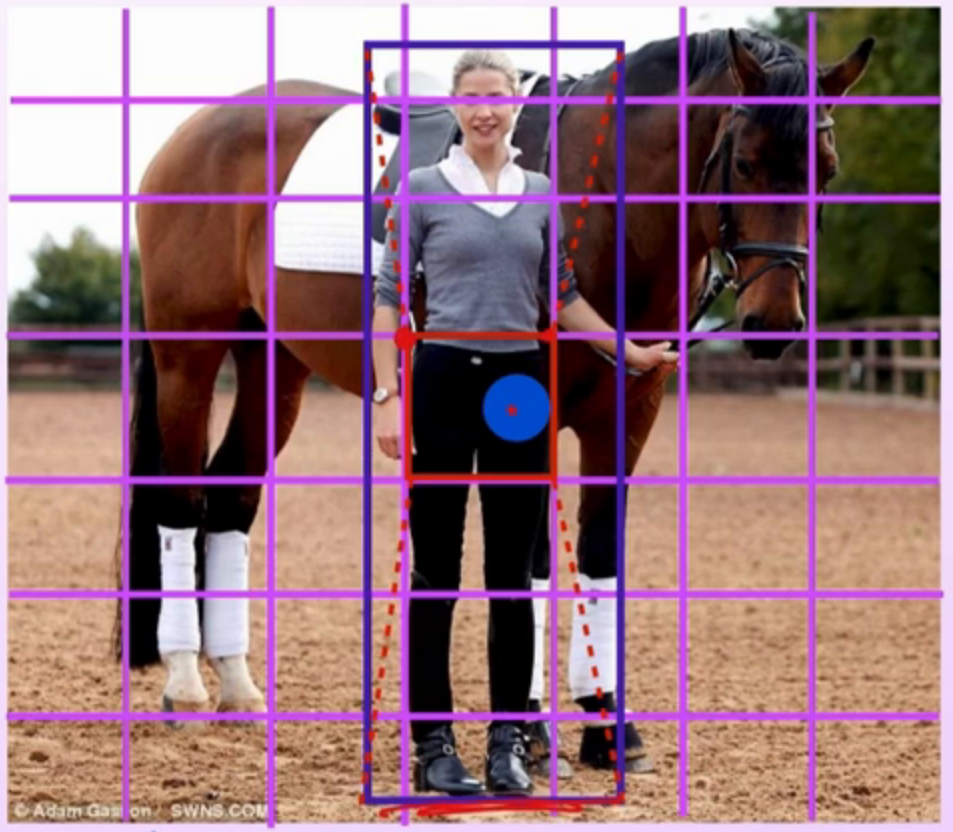
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





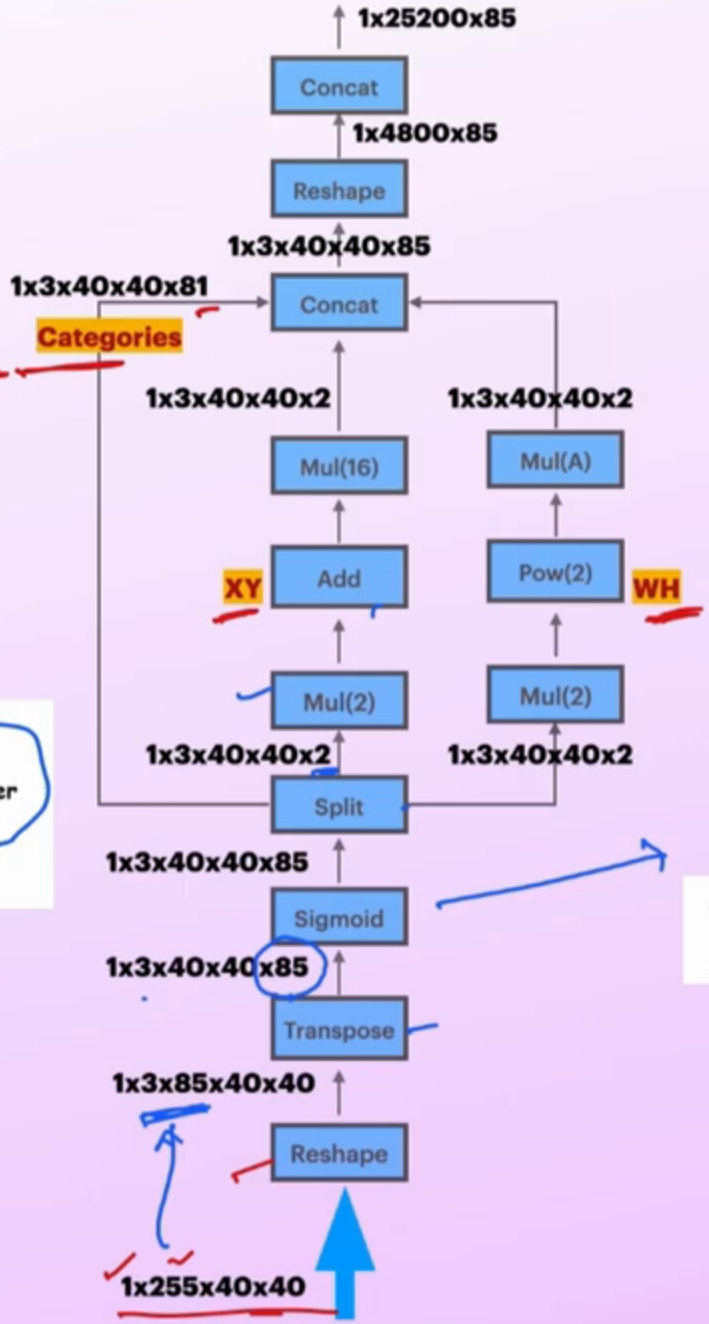


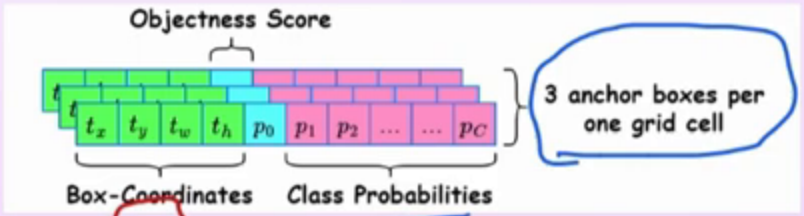
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

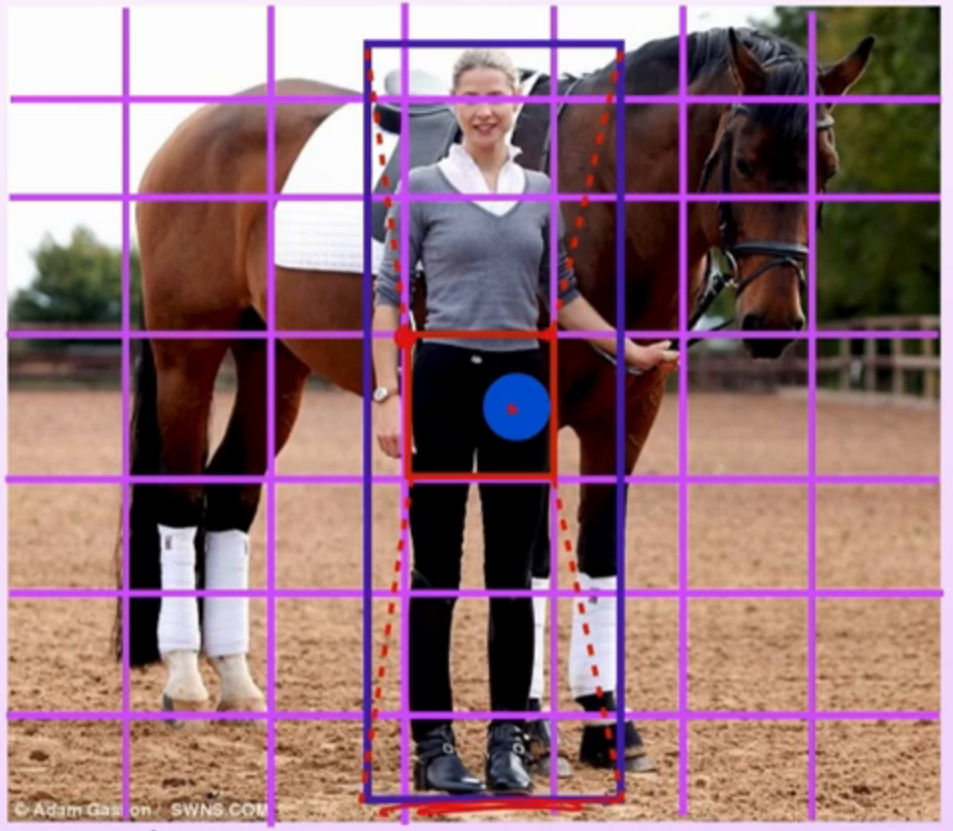
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





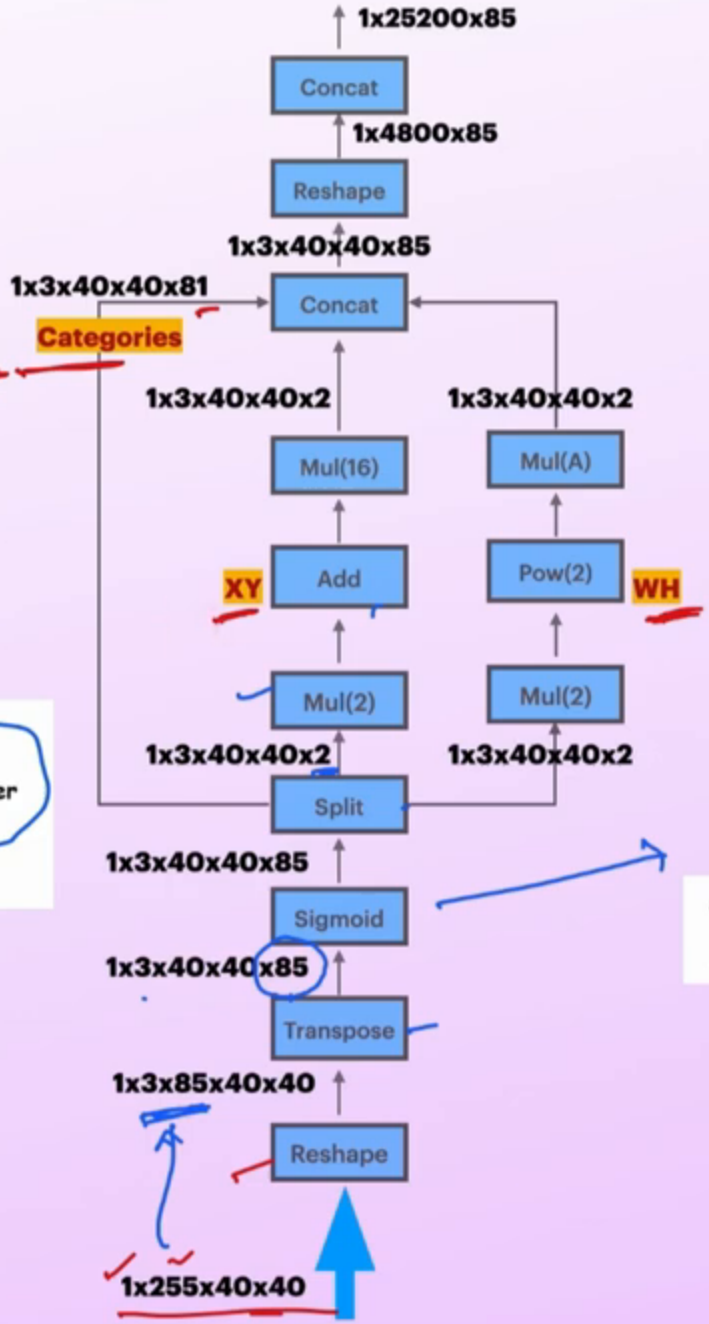


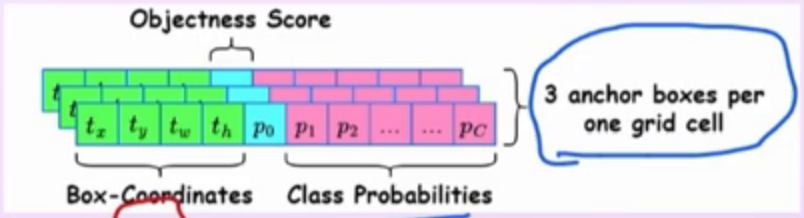
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

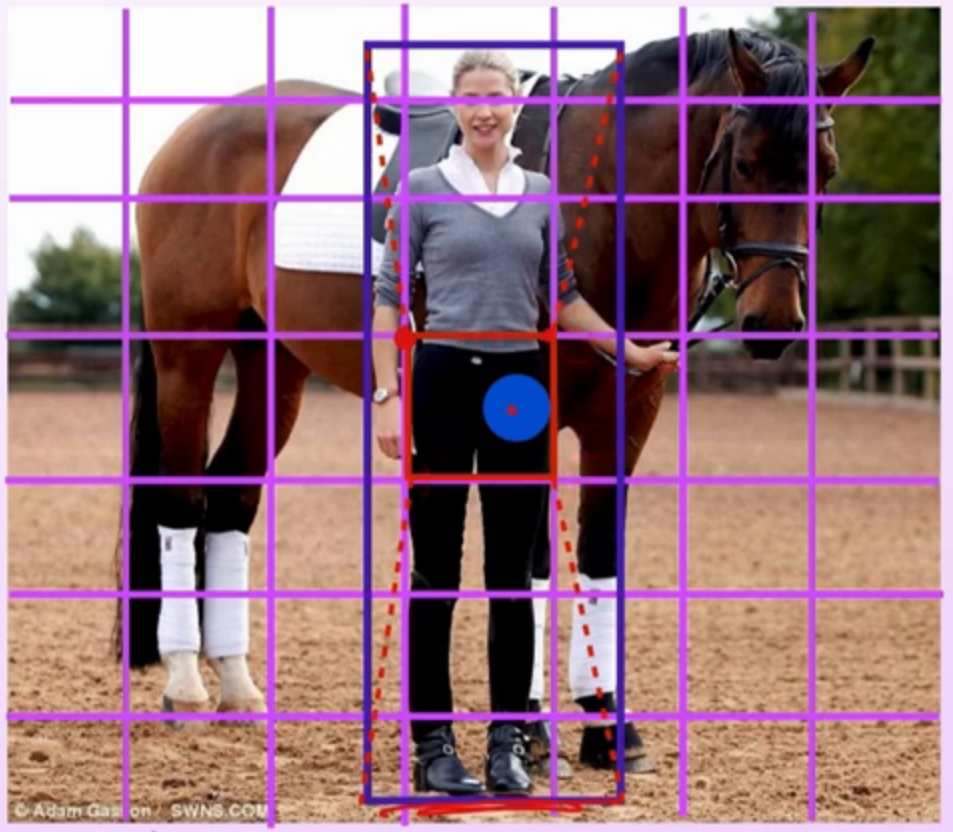
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





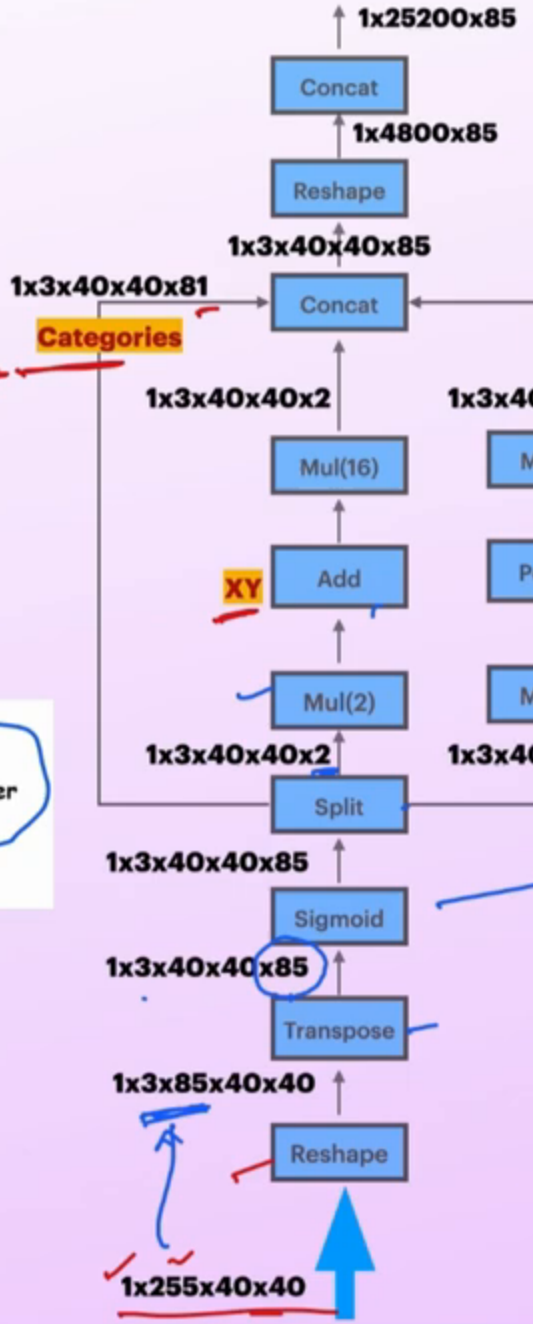


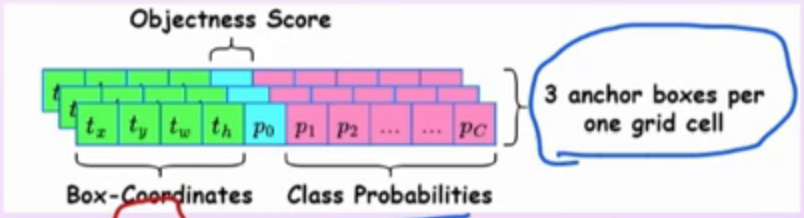
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

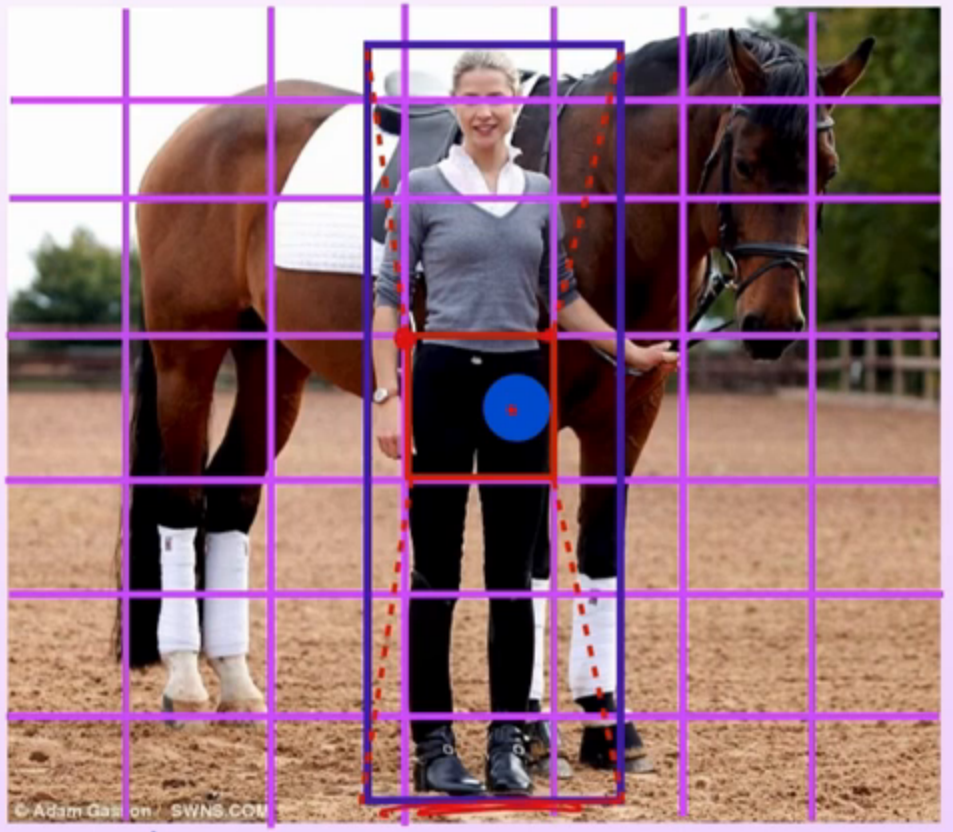
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





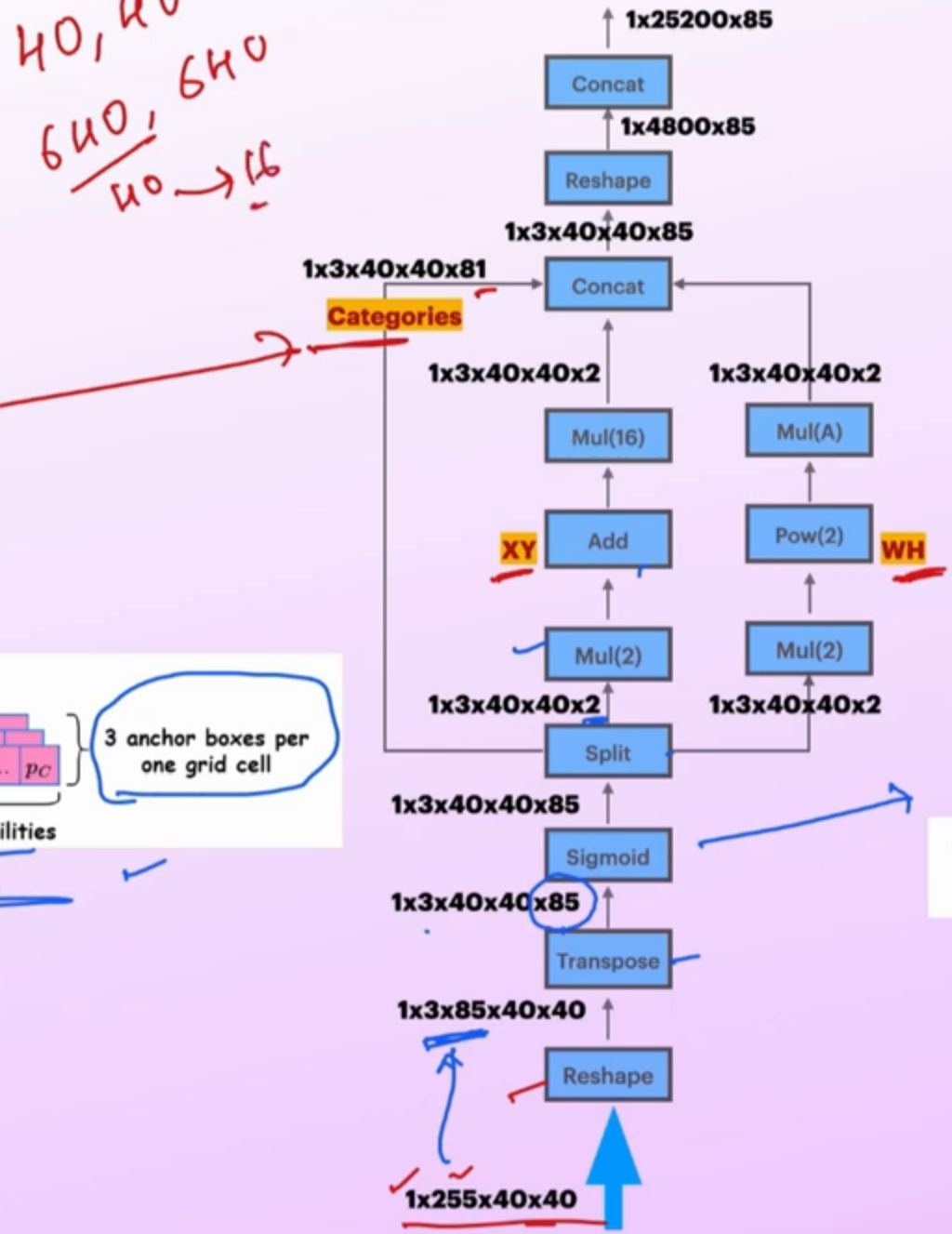


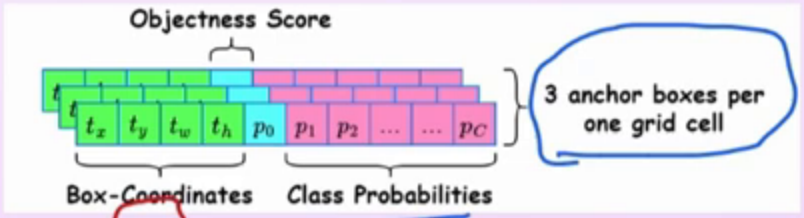
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

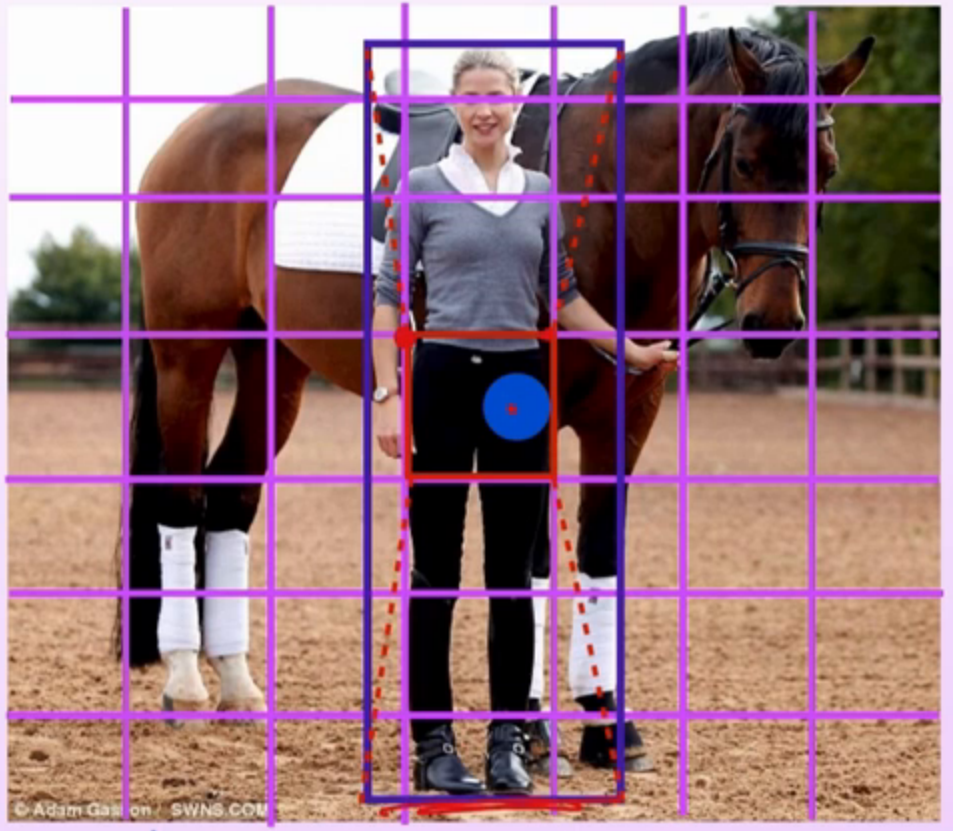
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





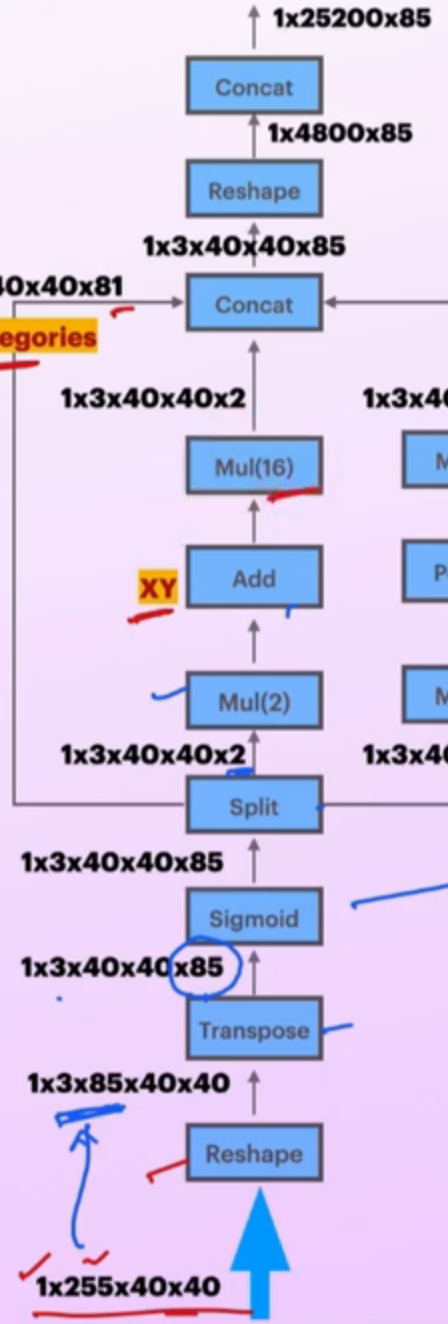


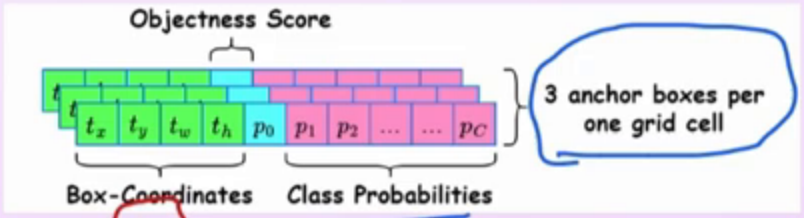
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

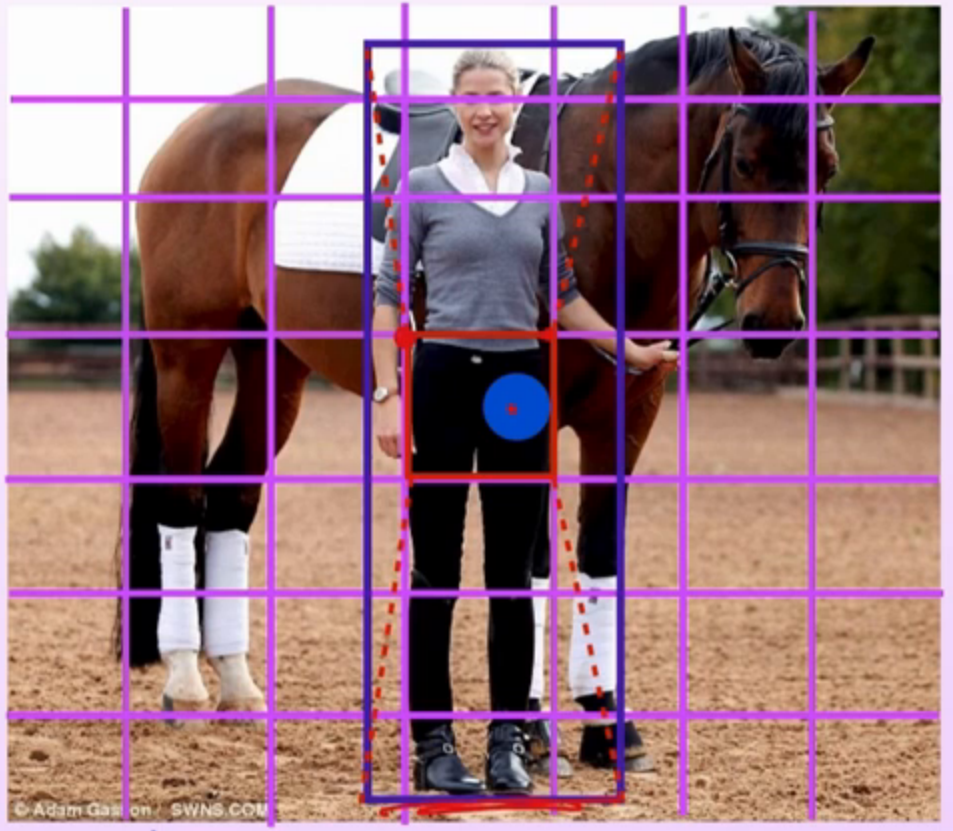
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





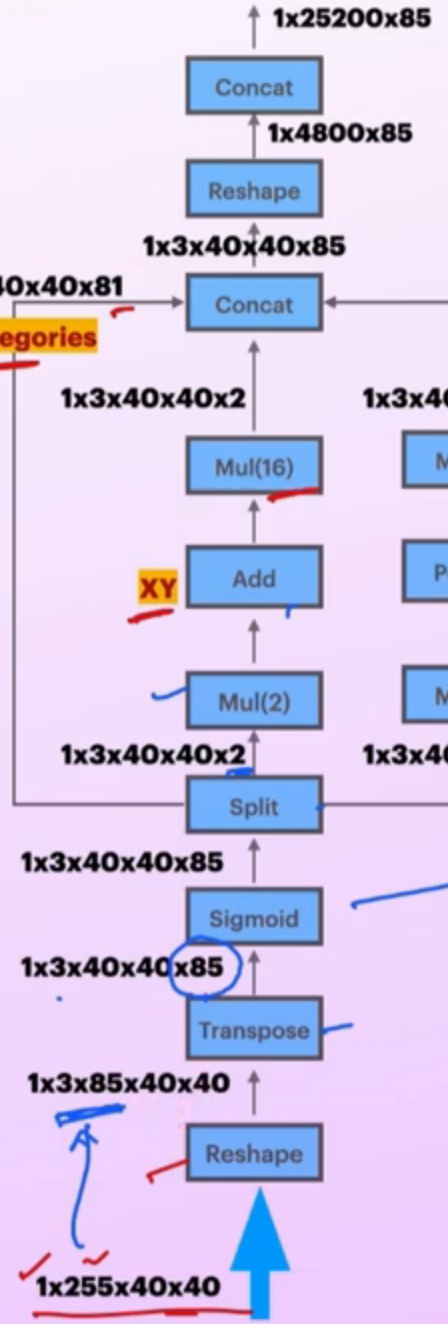


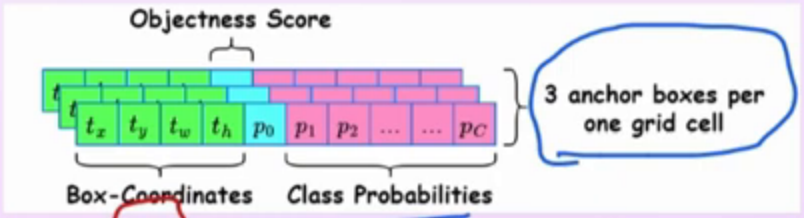
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

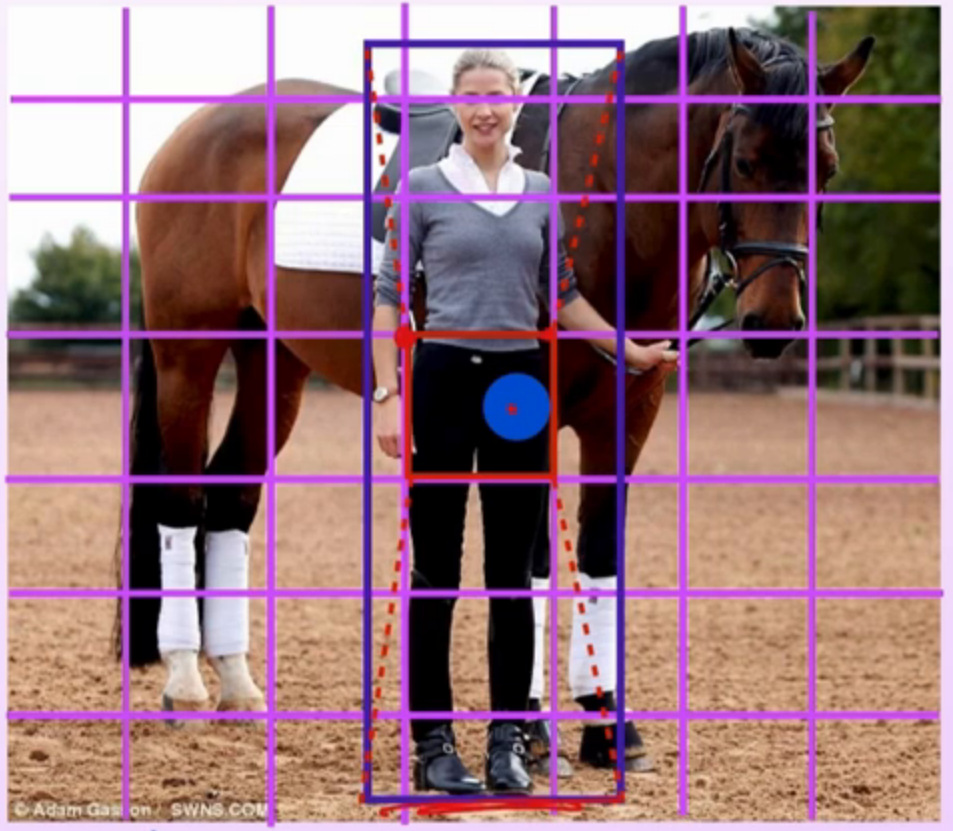
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





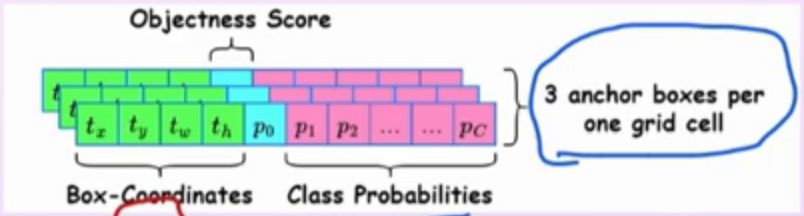


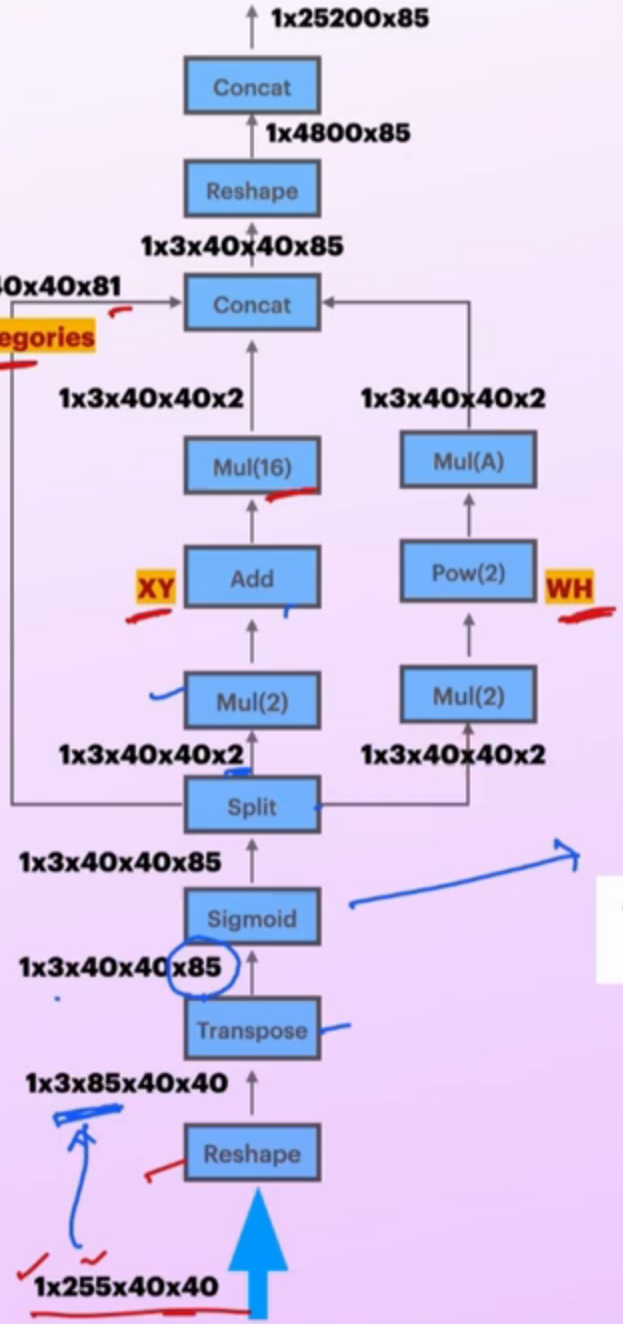
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

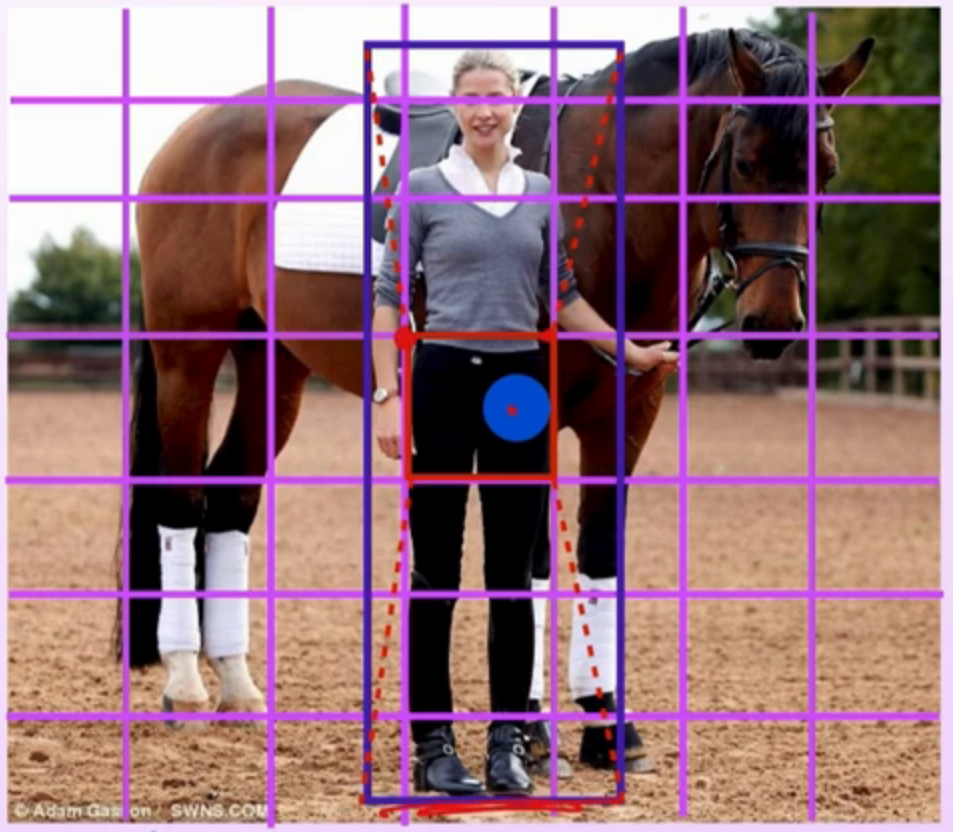
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





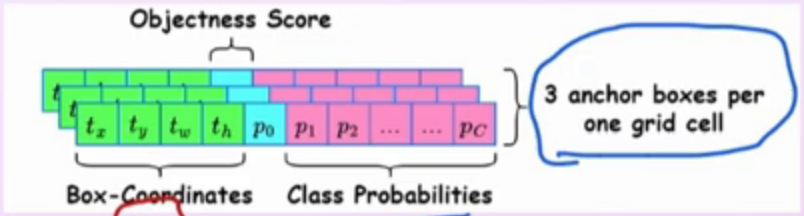


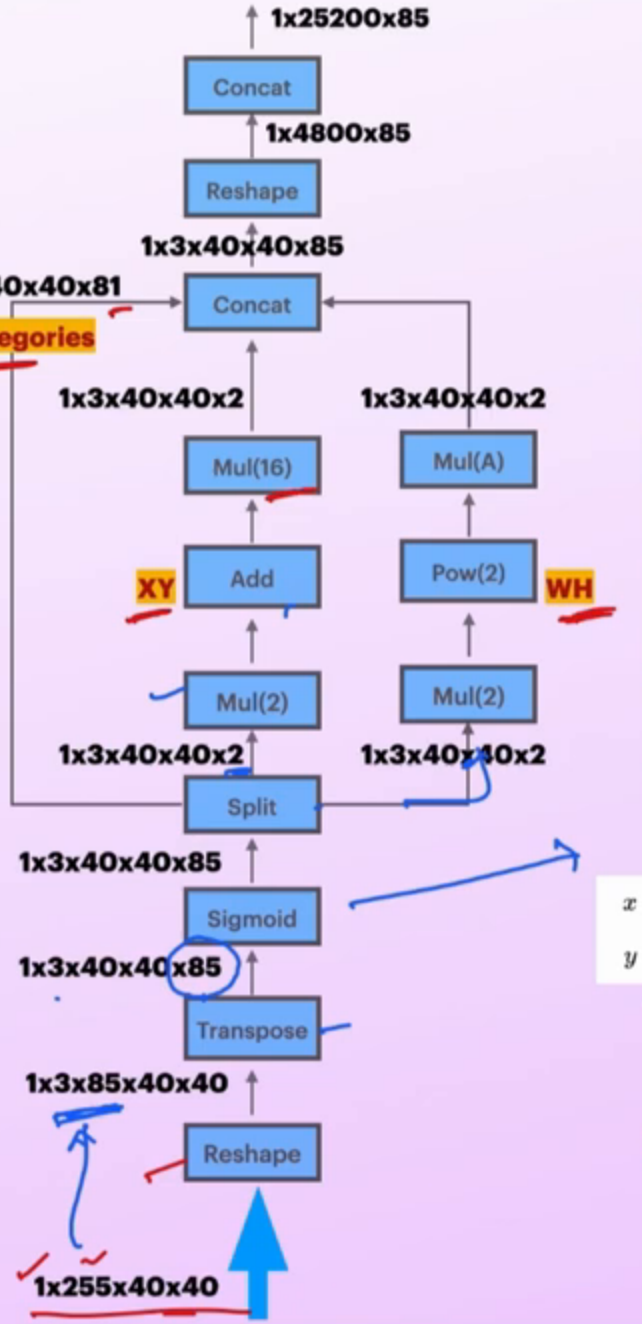
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

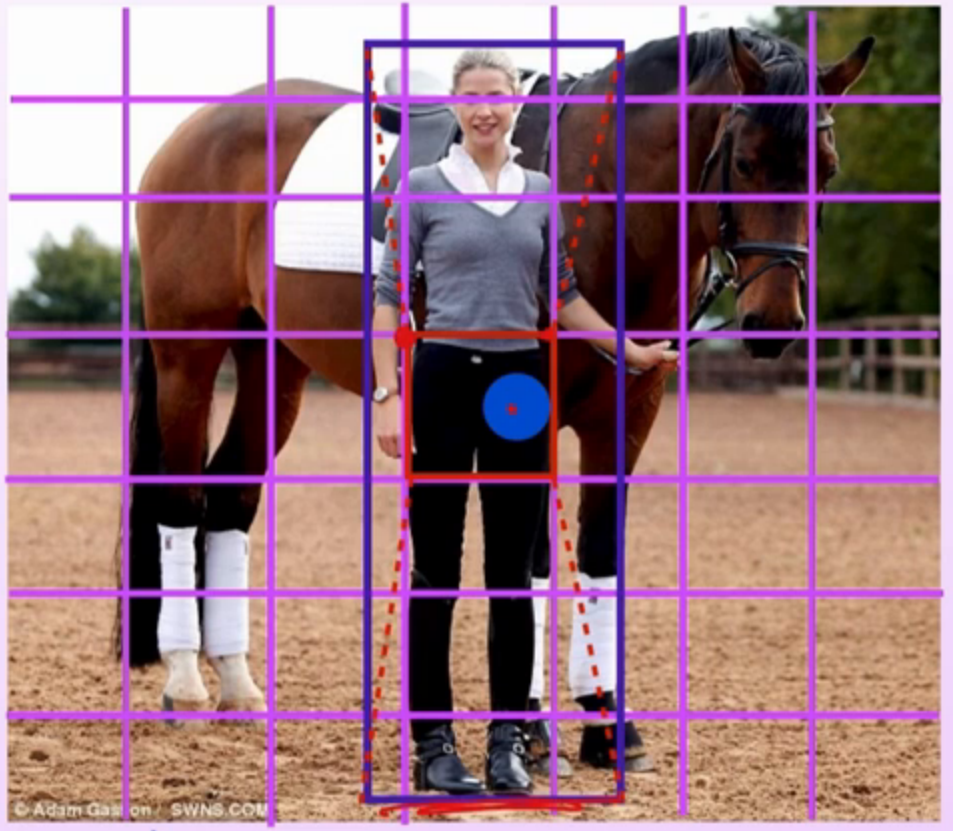
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





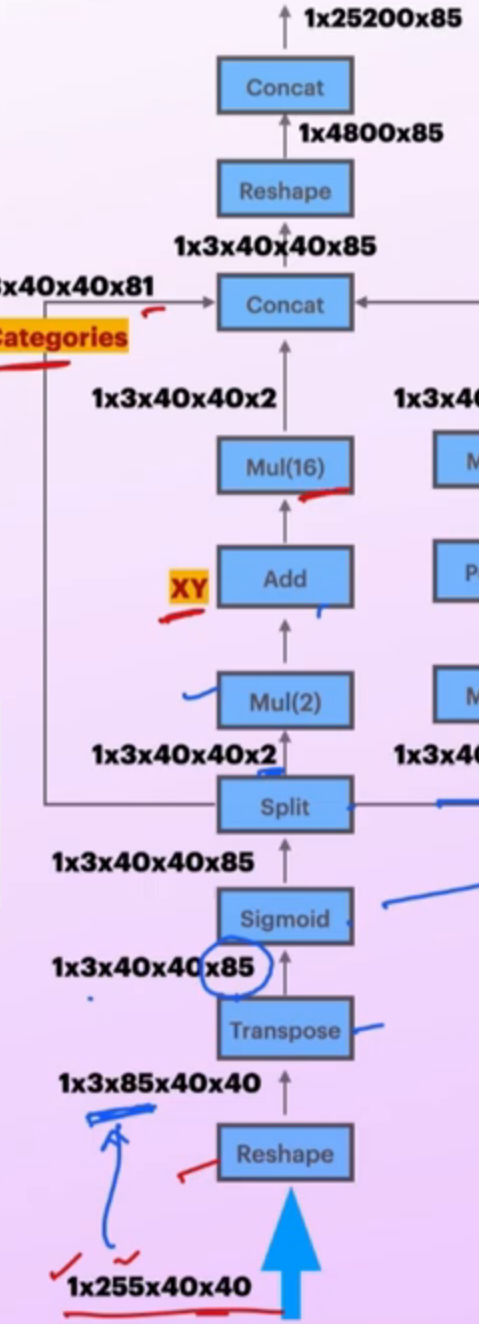


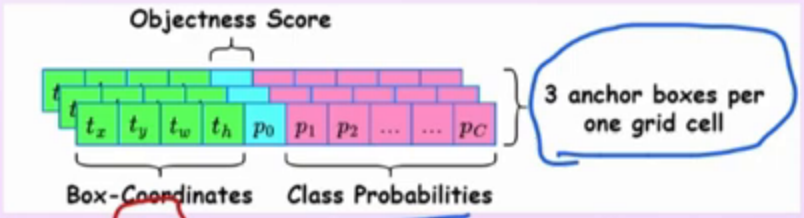
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

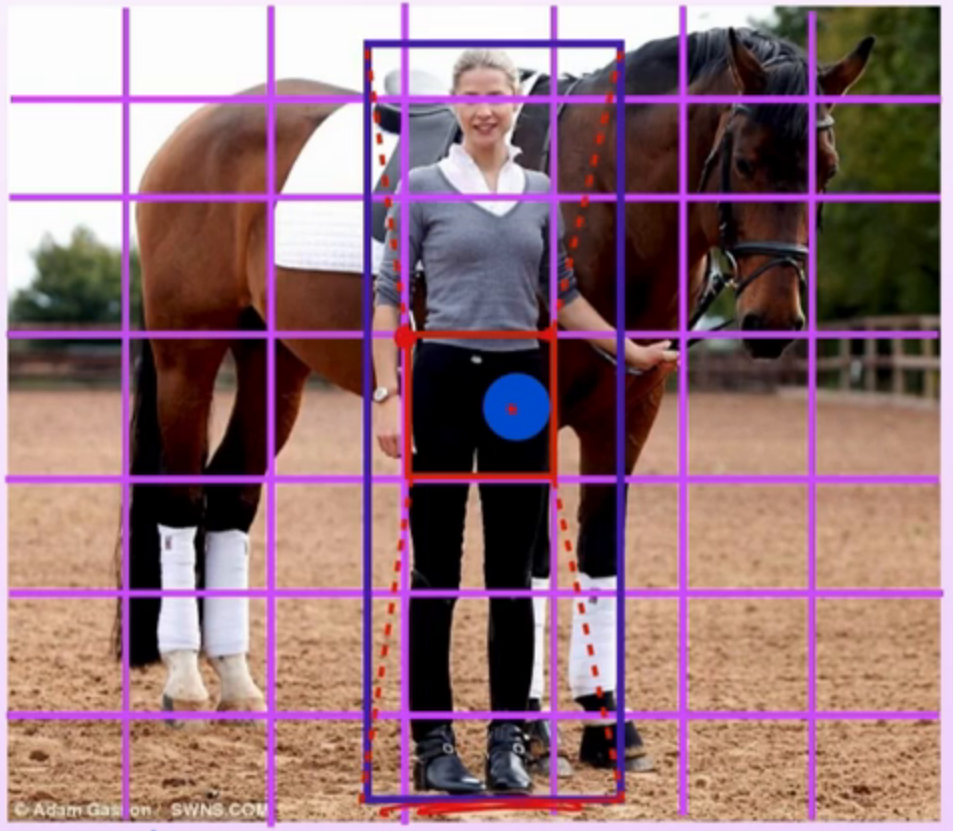
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





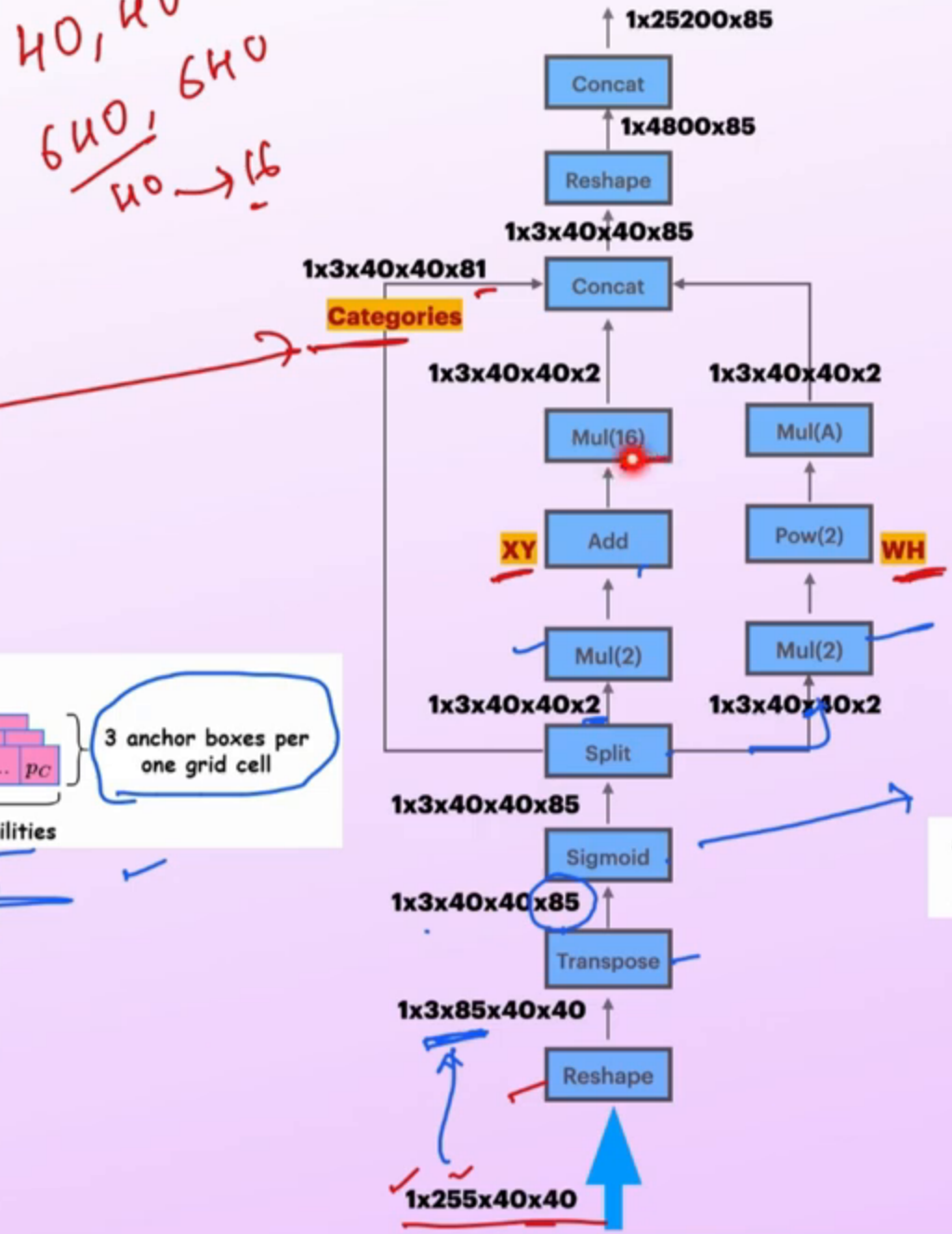


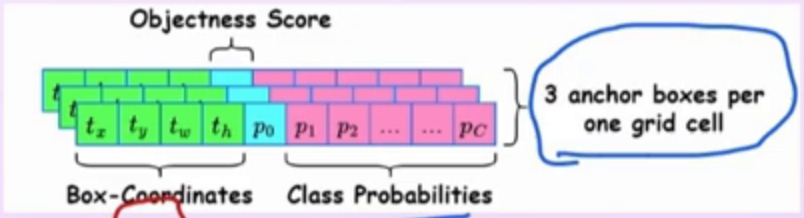
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

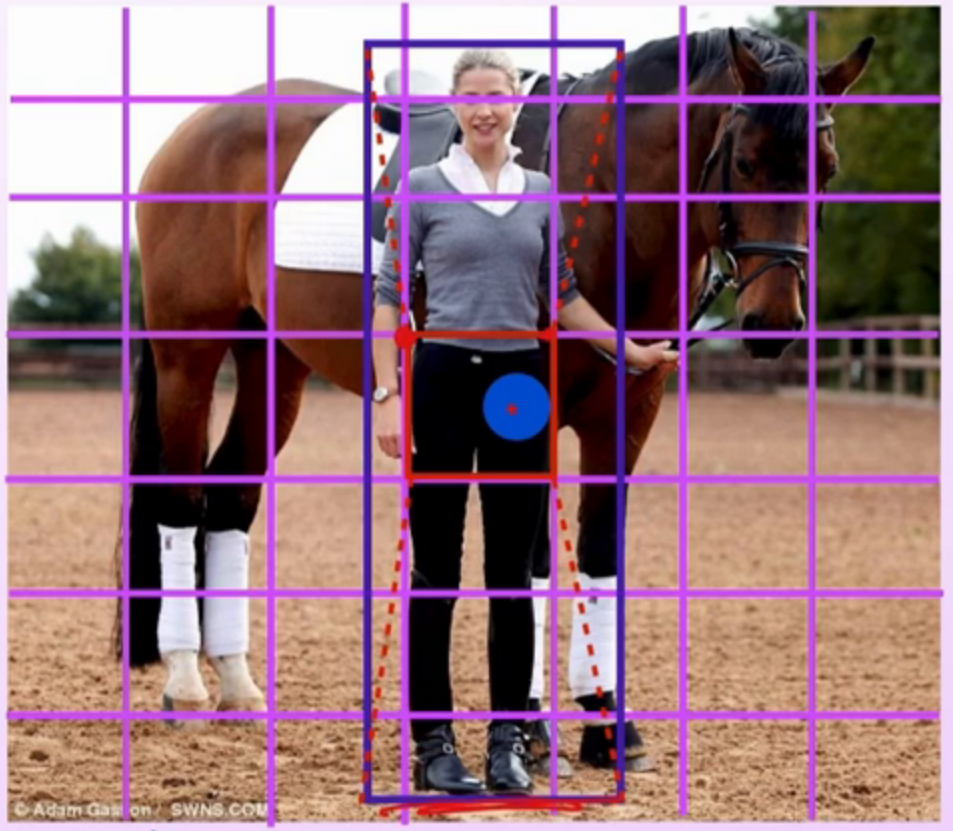
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





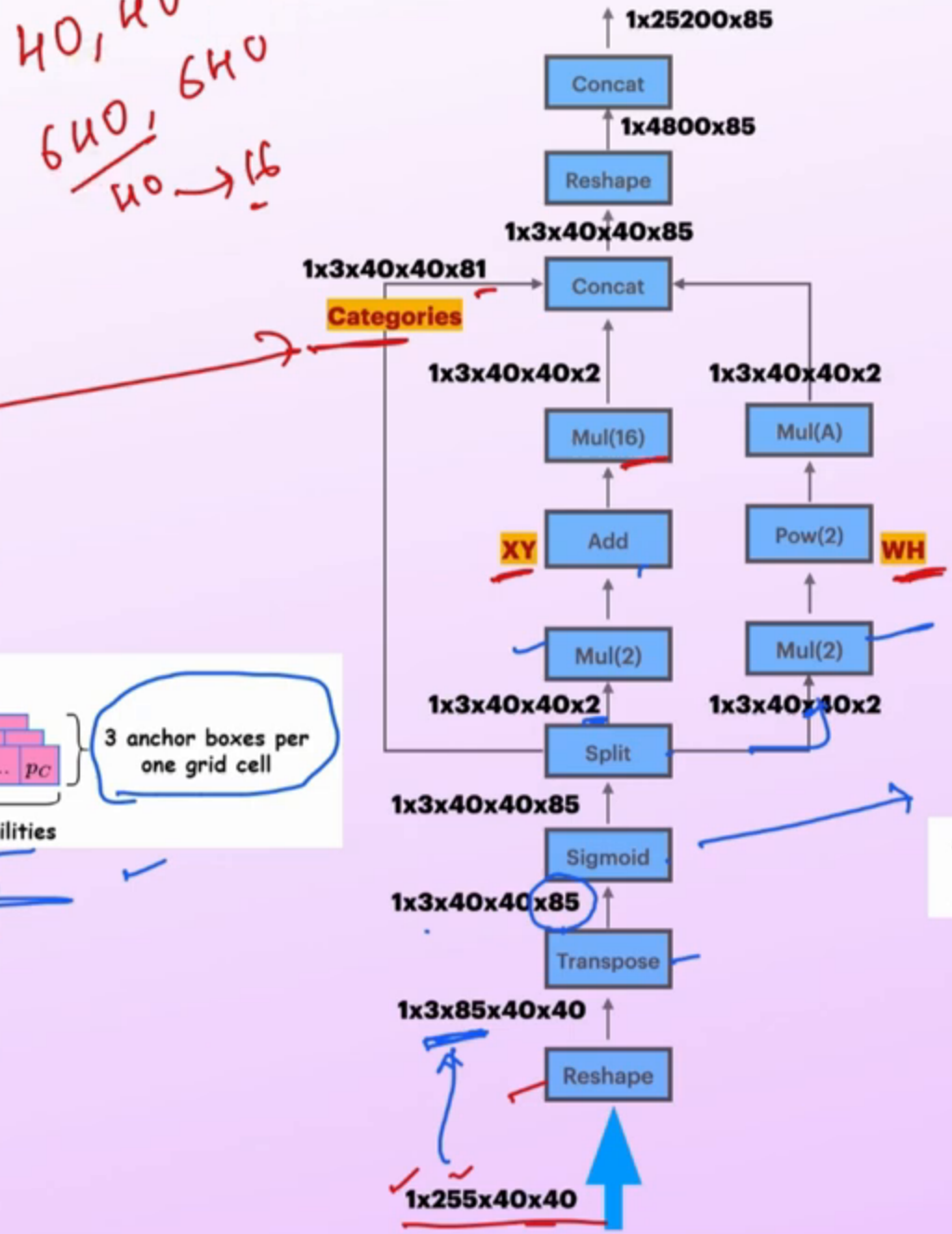


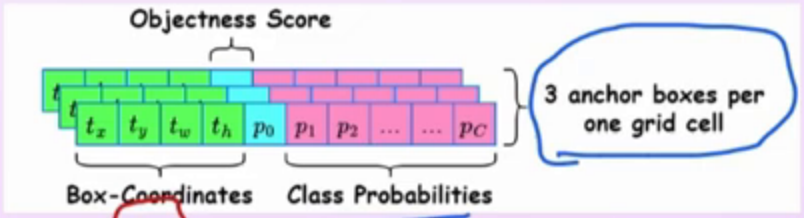
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

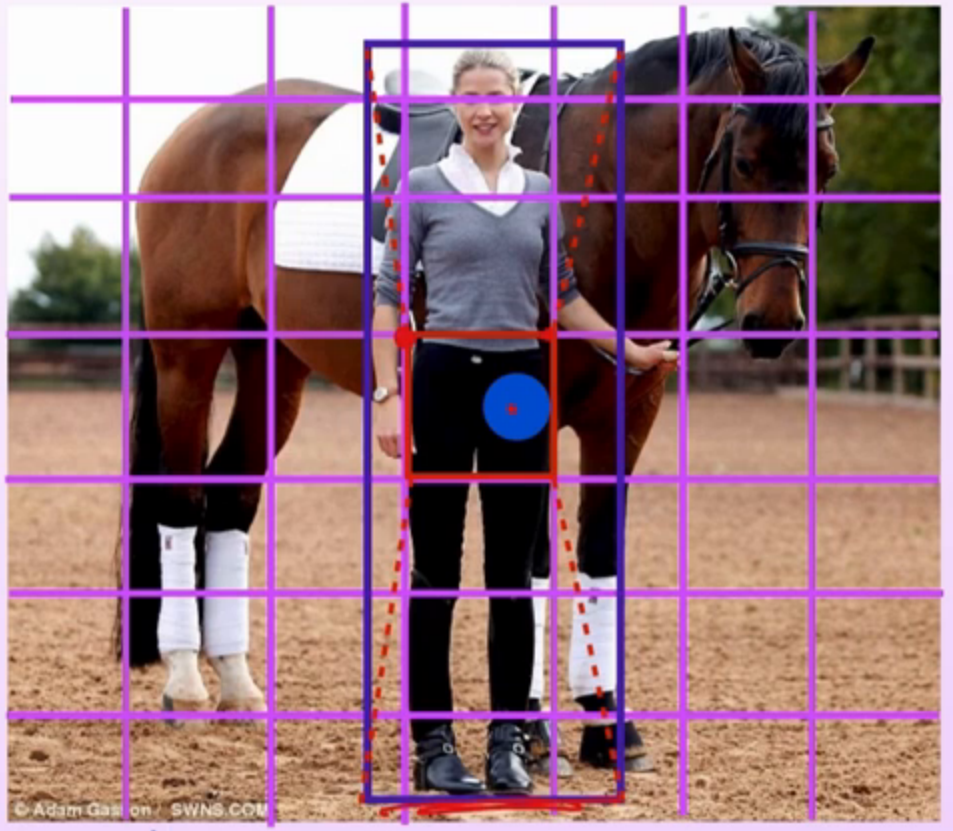
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





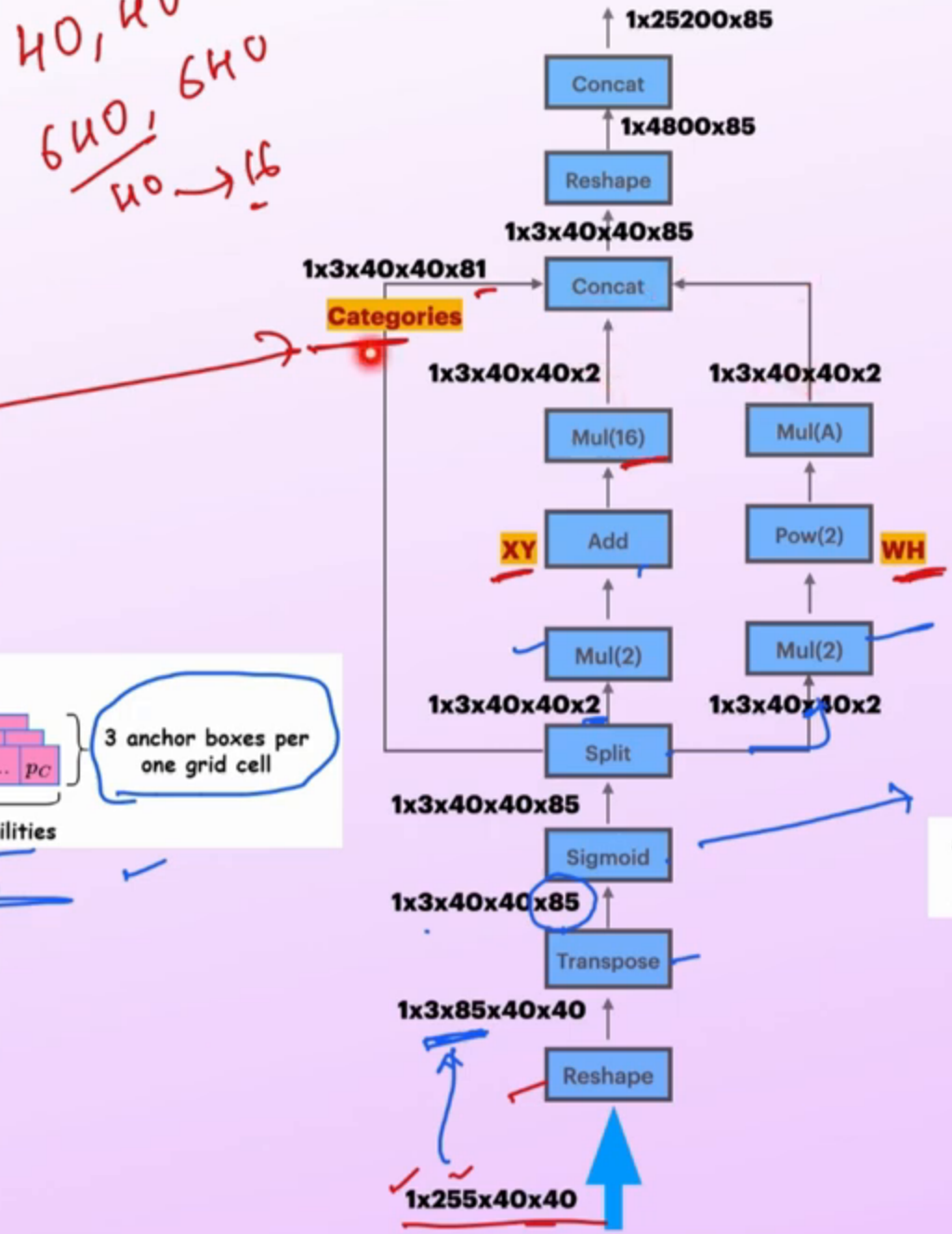


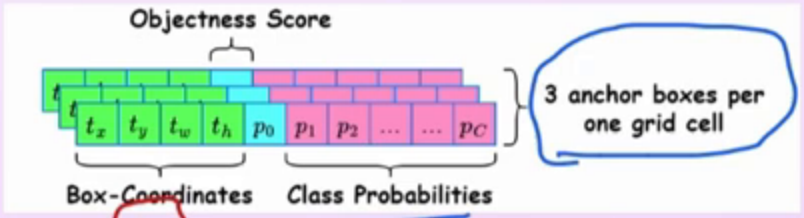
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

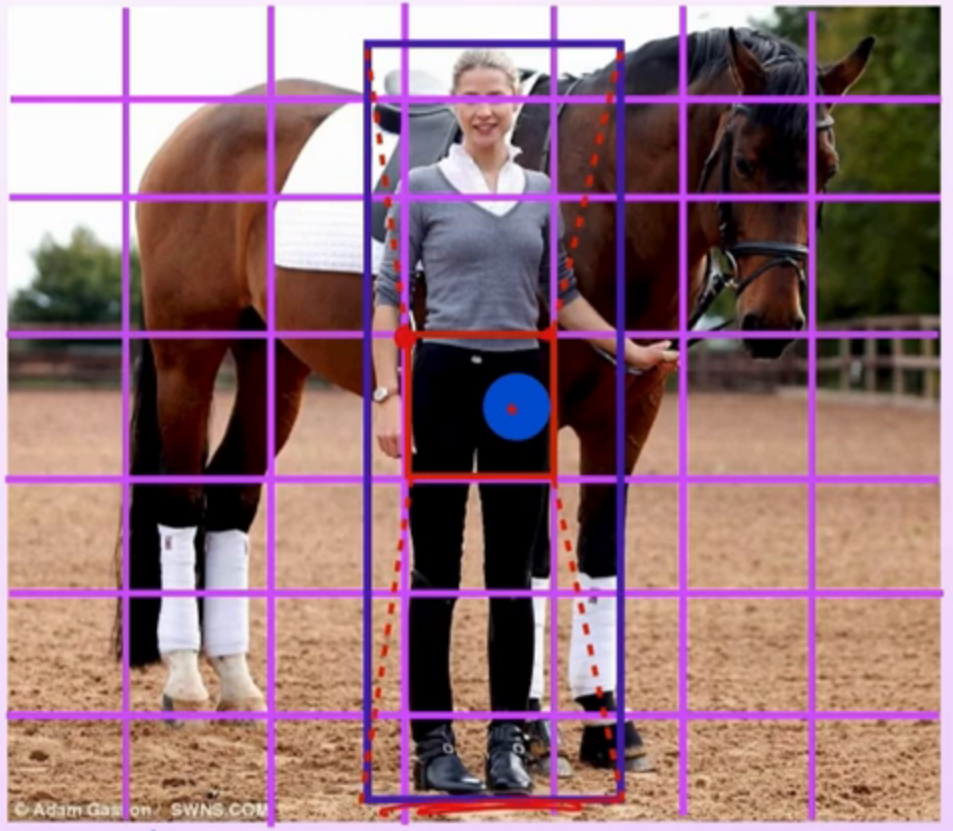
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





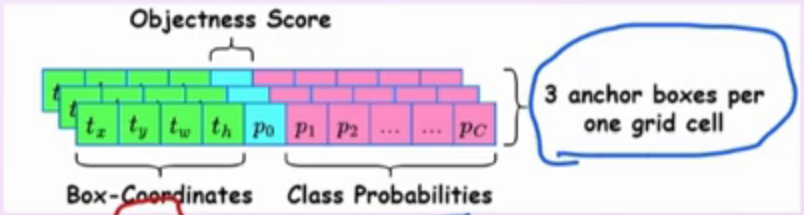


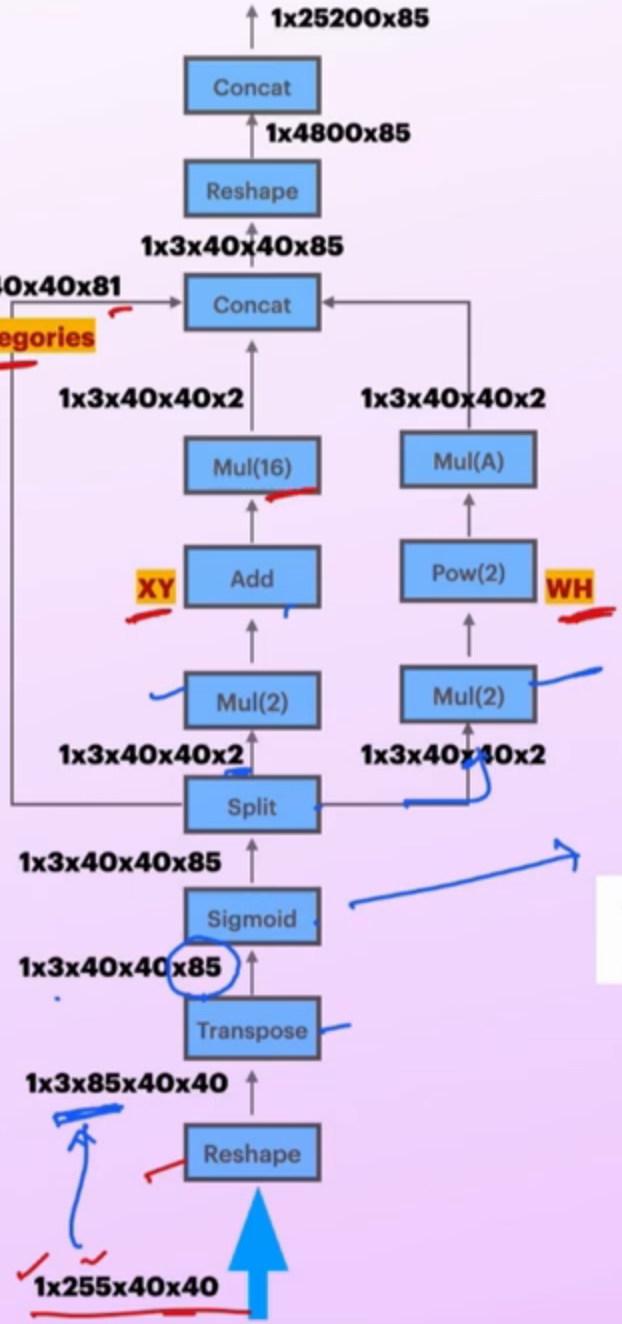
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

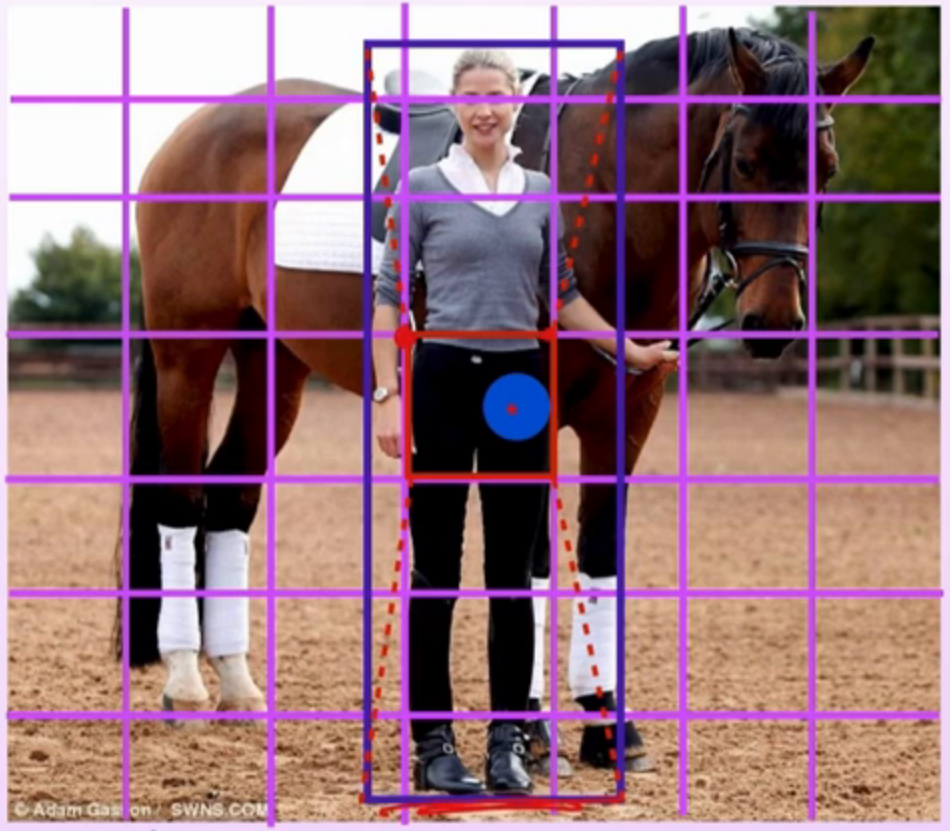
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





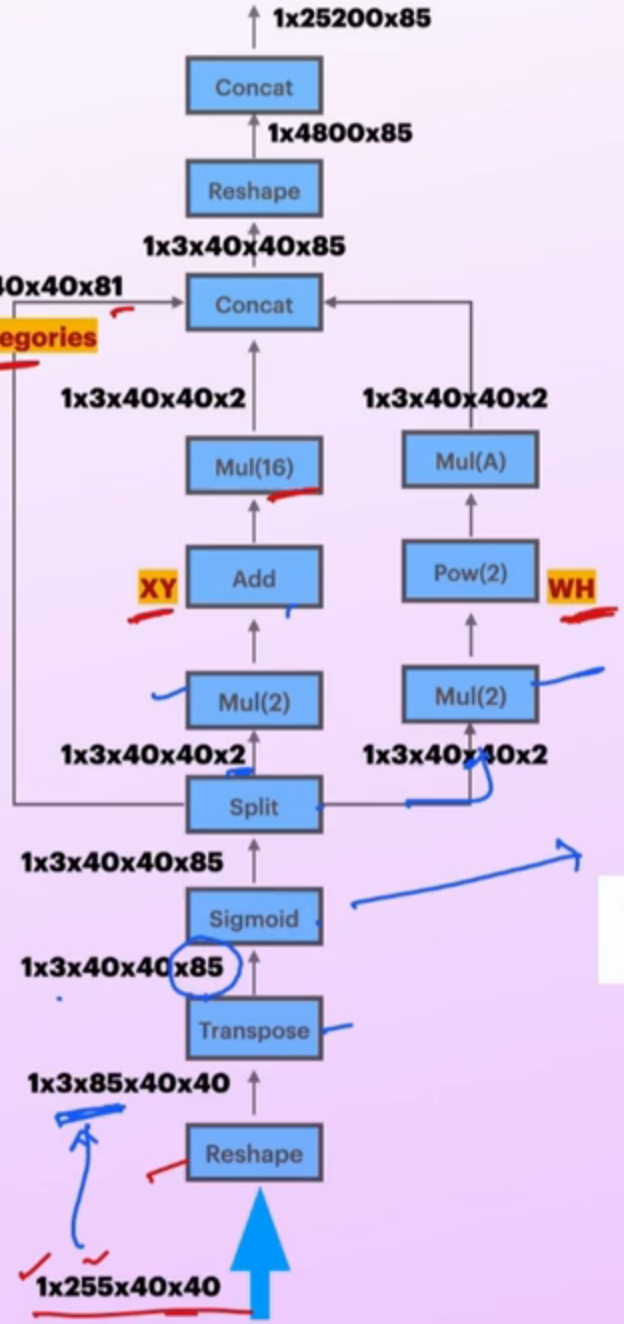


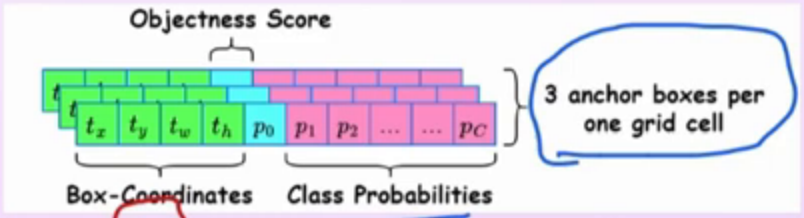
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

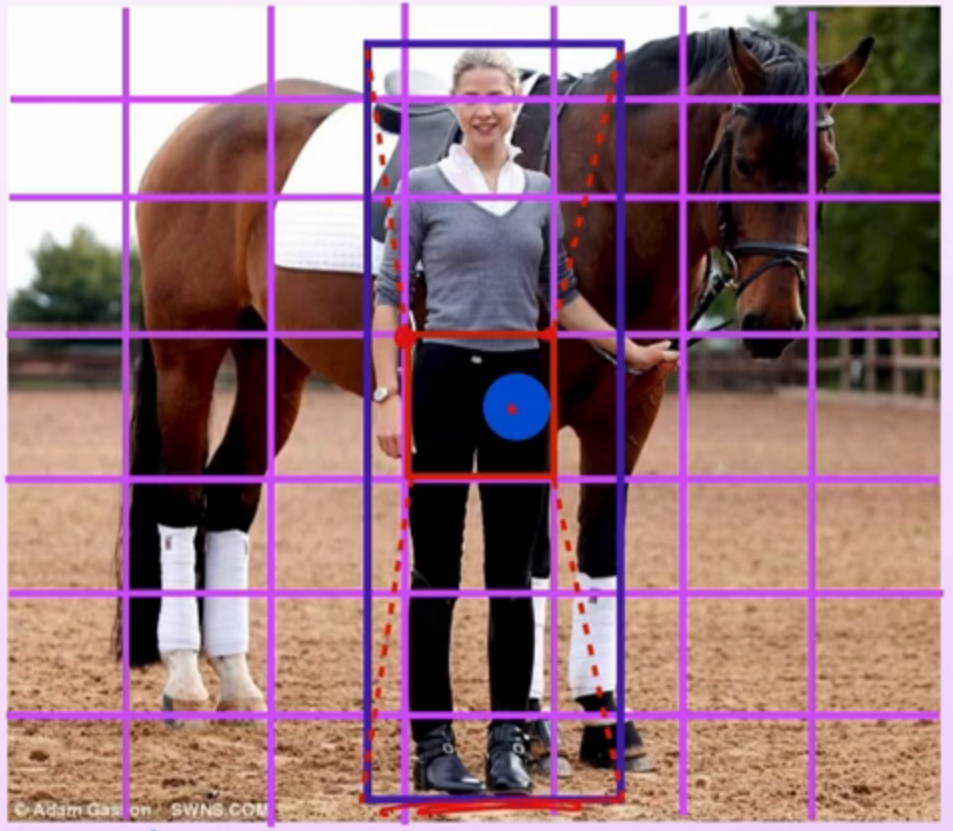
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





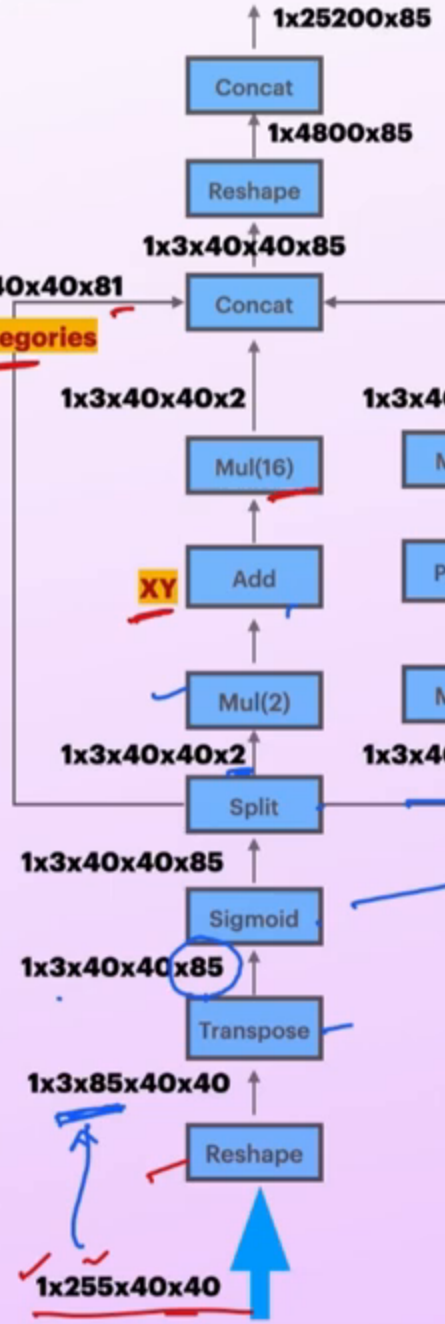


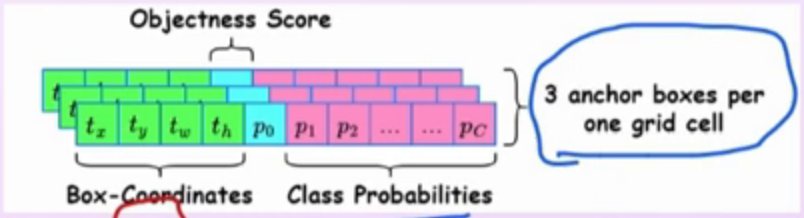
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

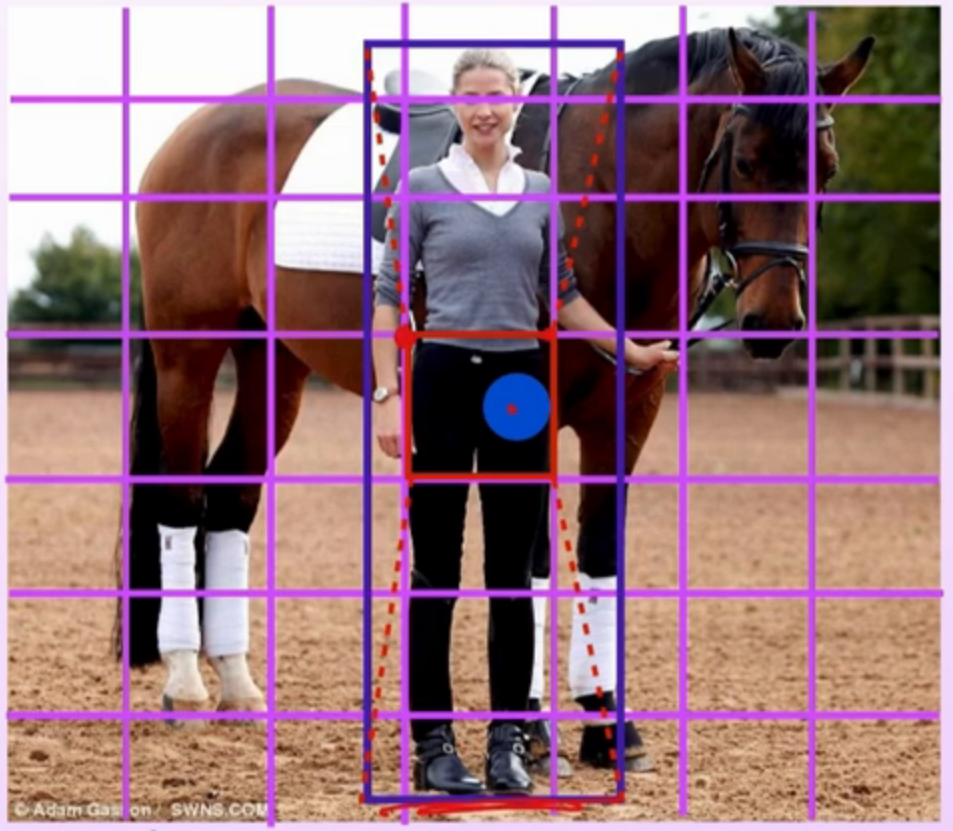
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





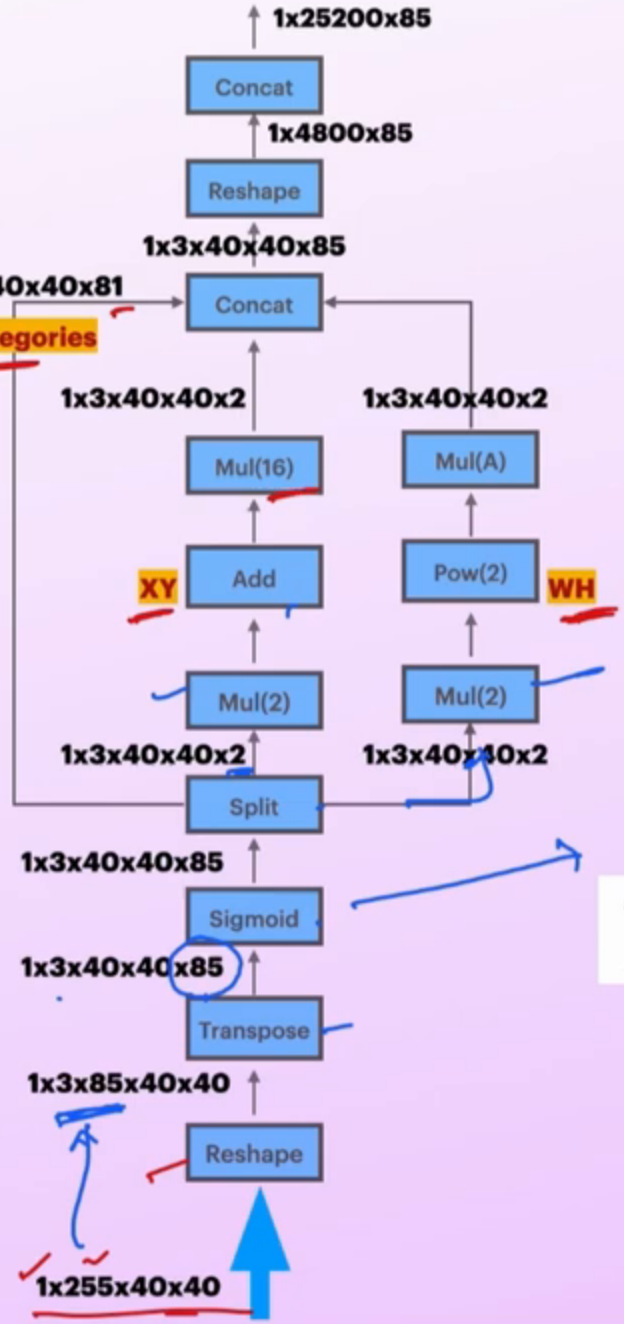


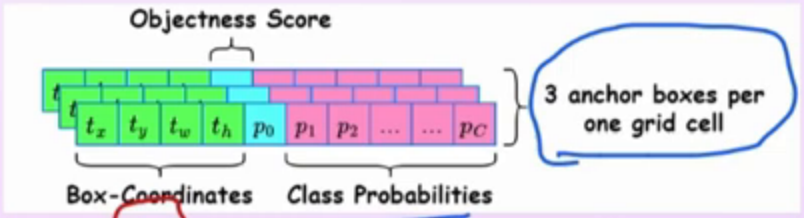
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

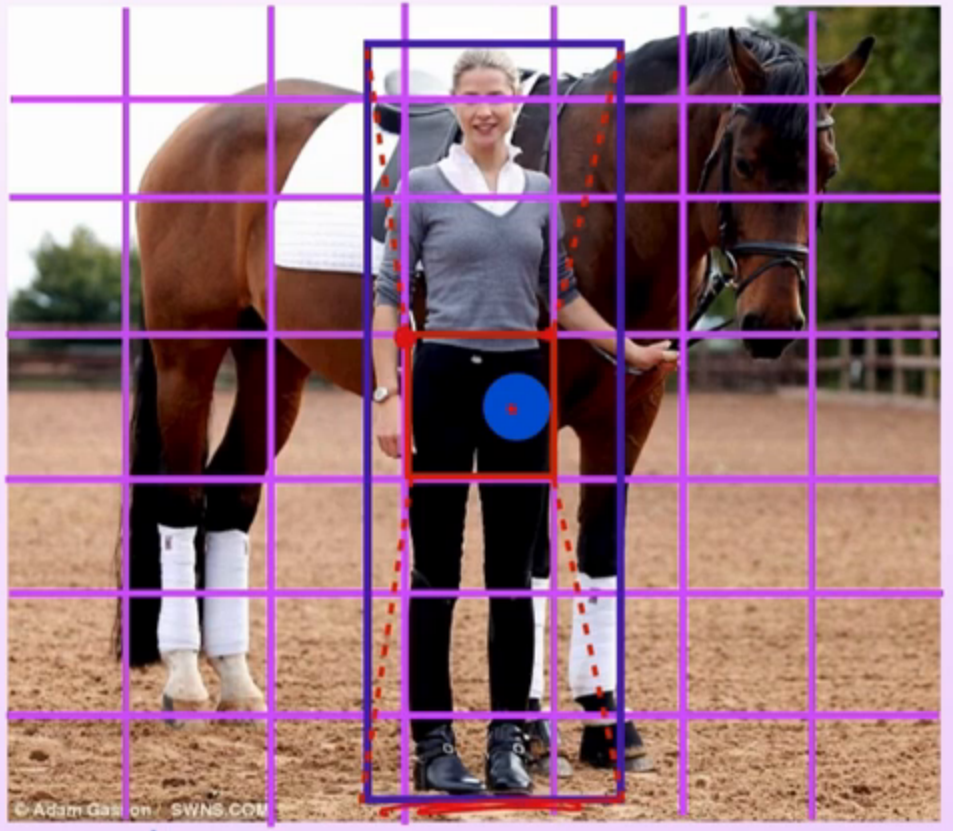
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





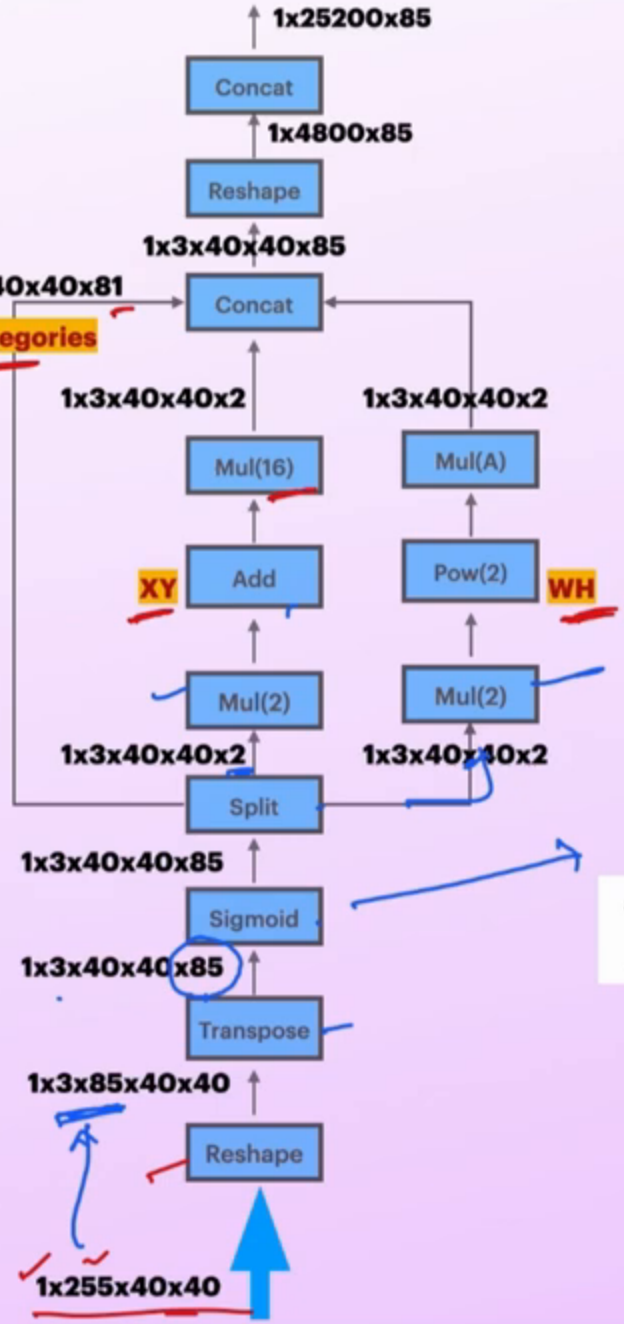


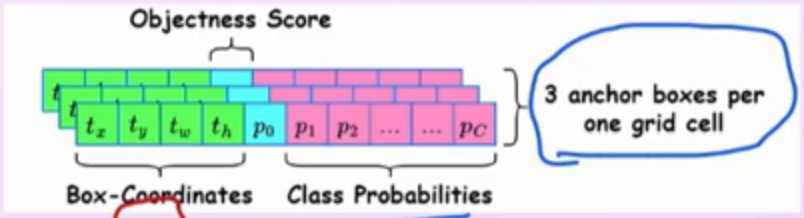
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

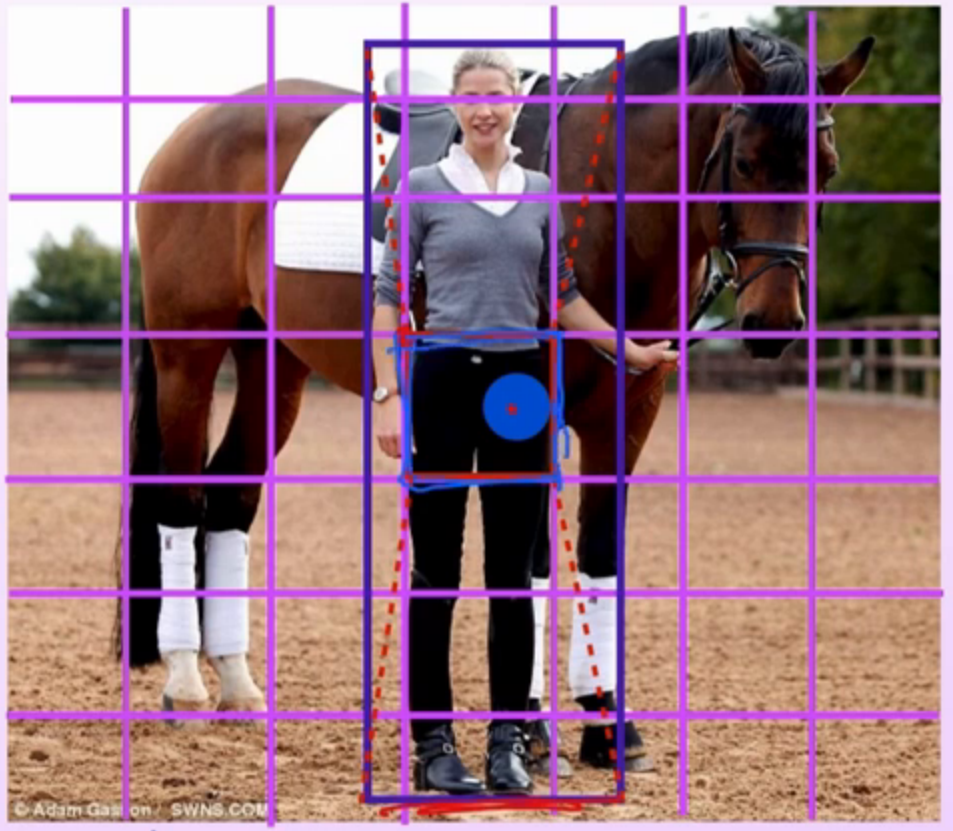
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





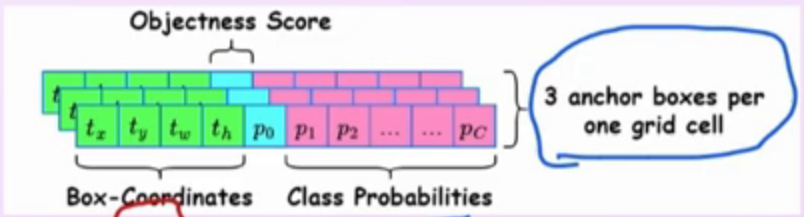


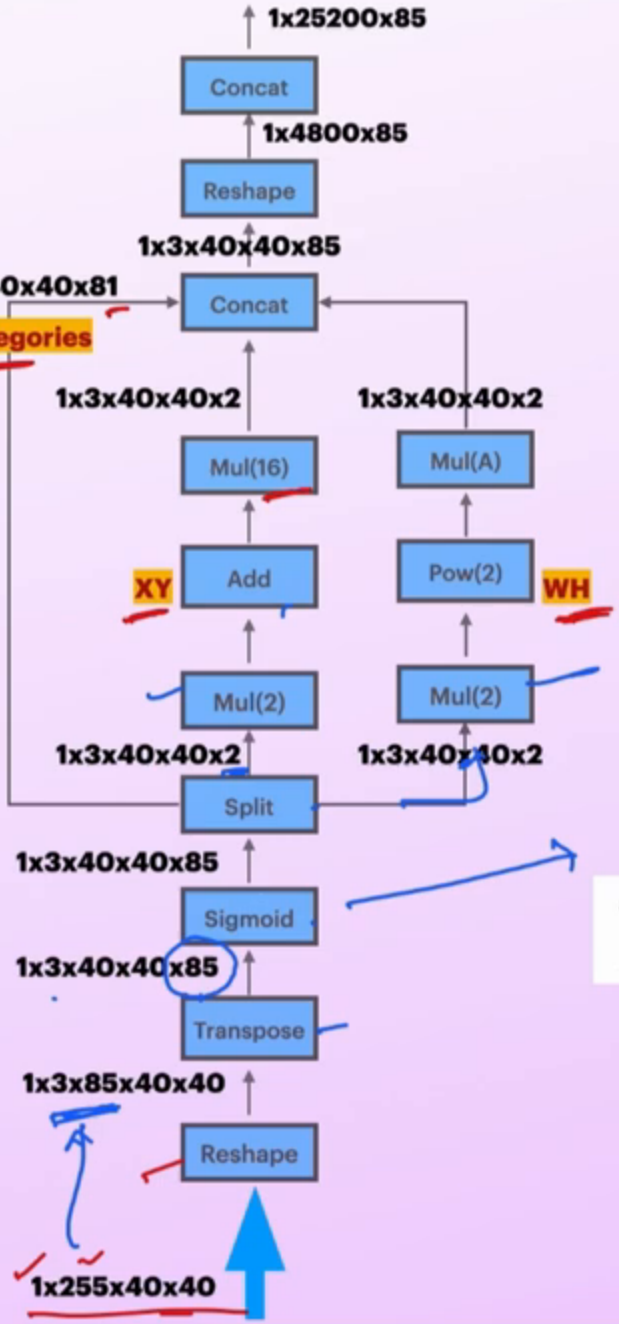
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

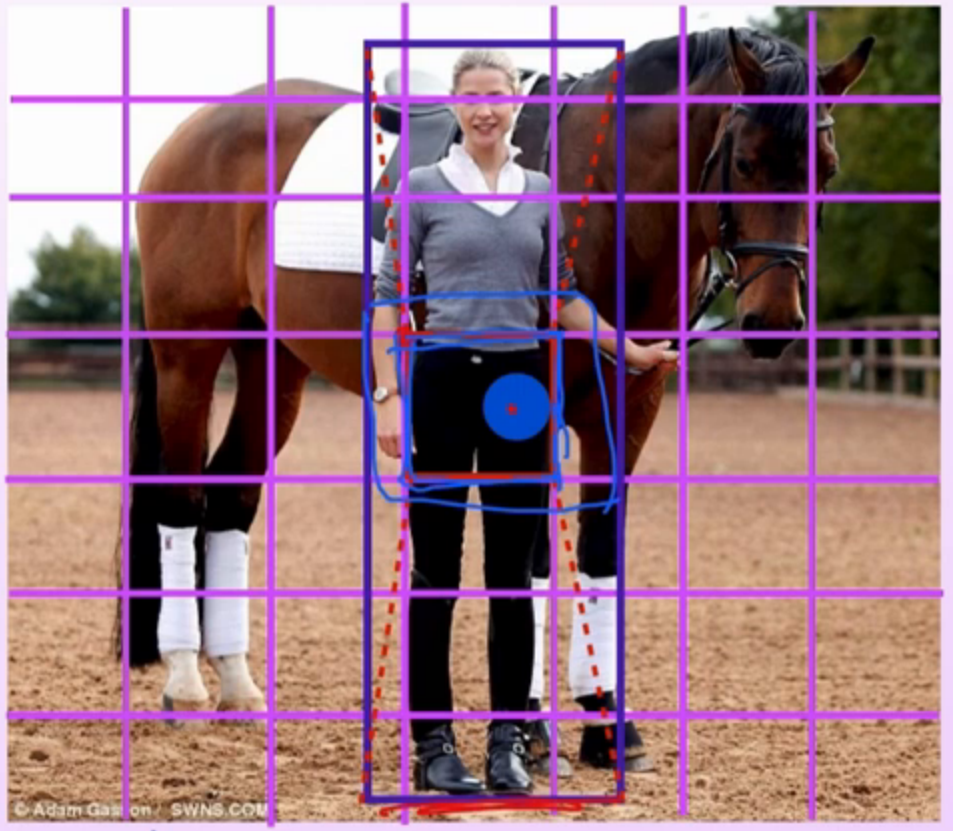
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





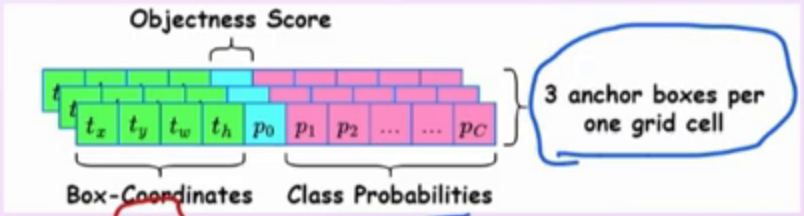


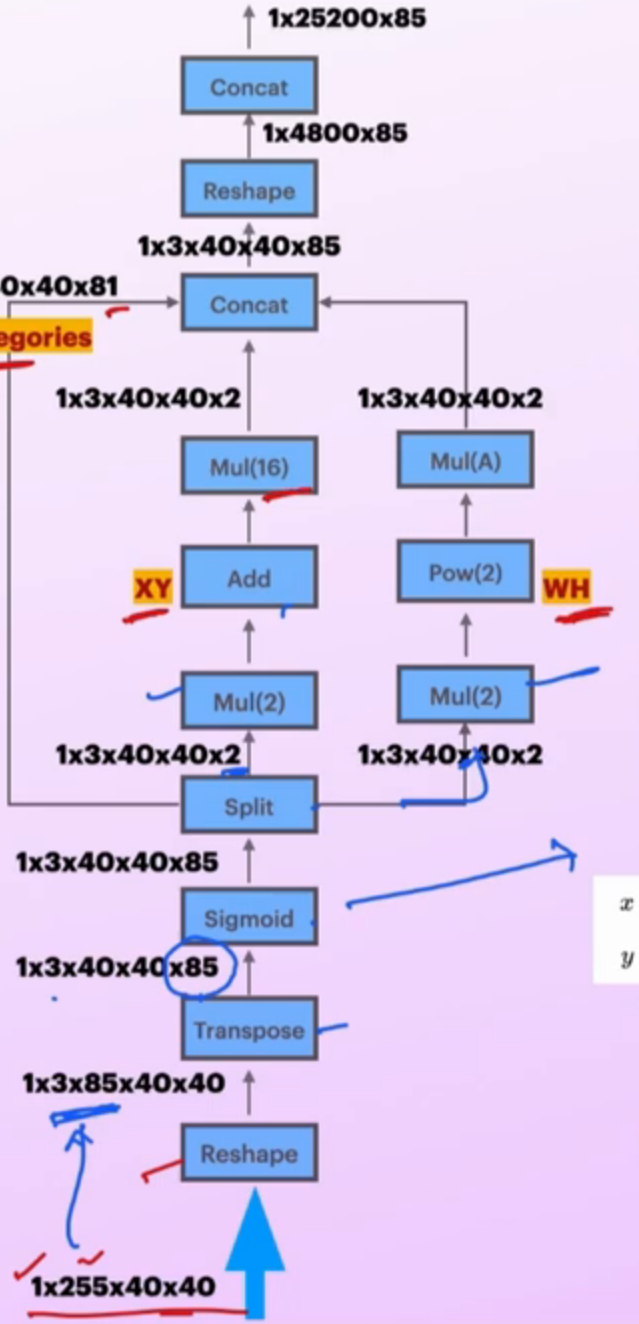
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

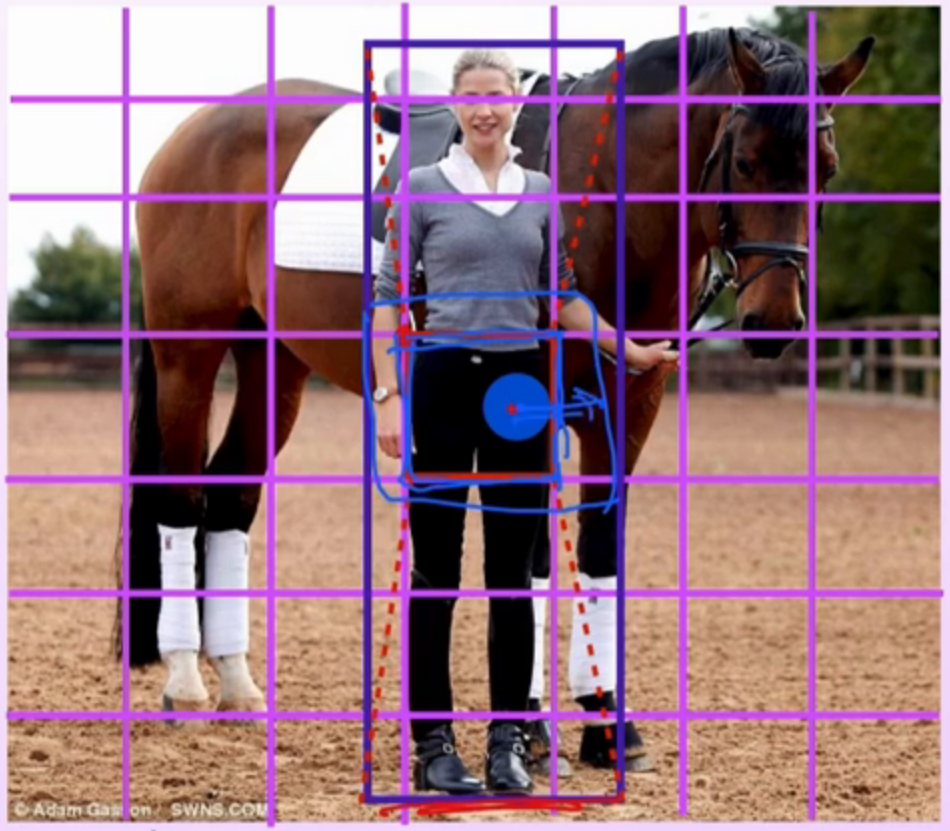
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing



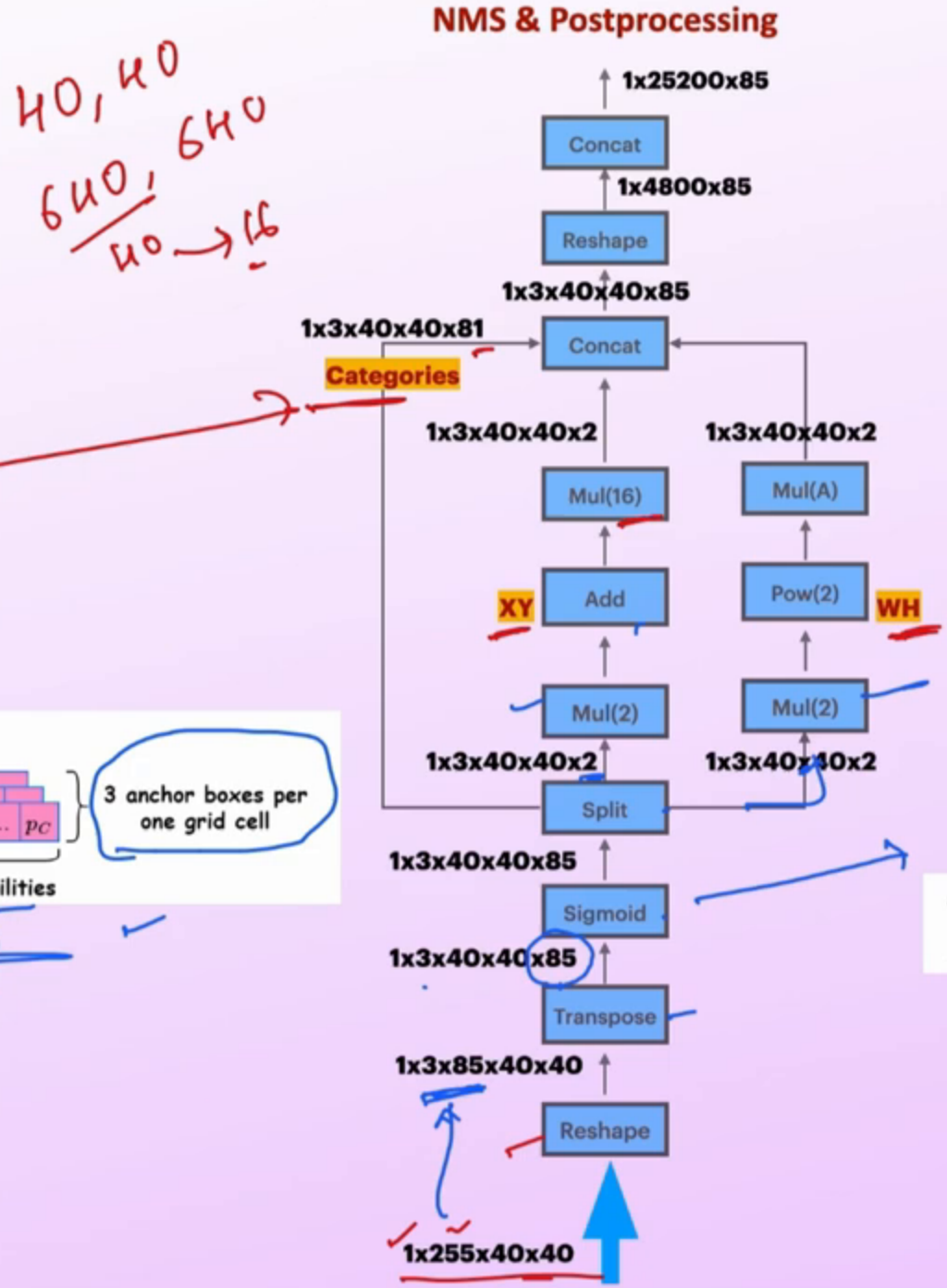


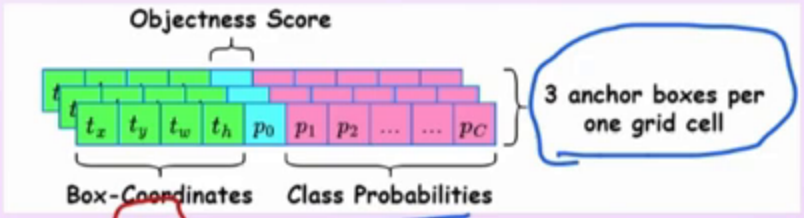


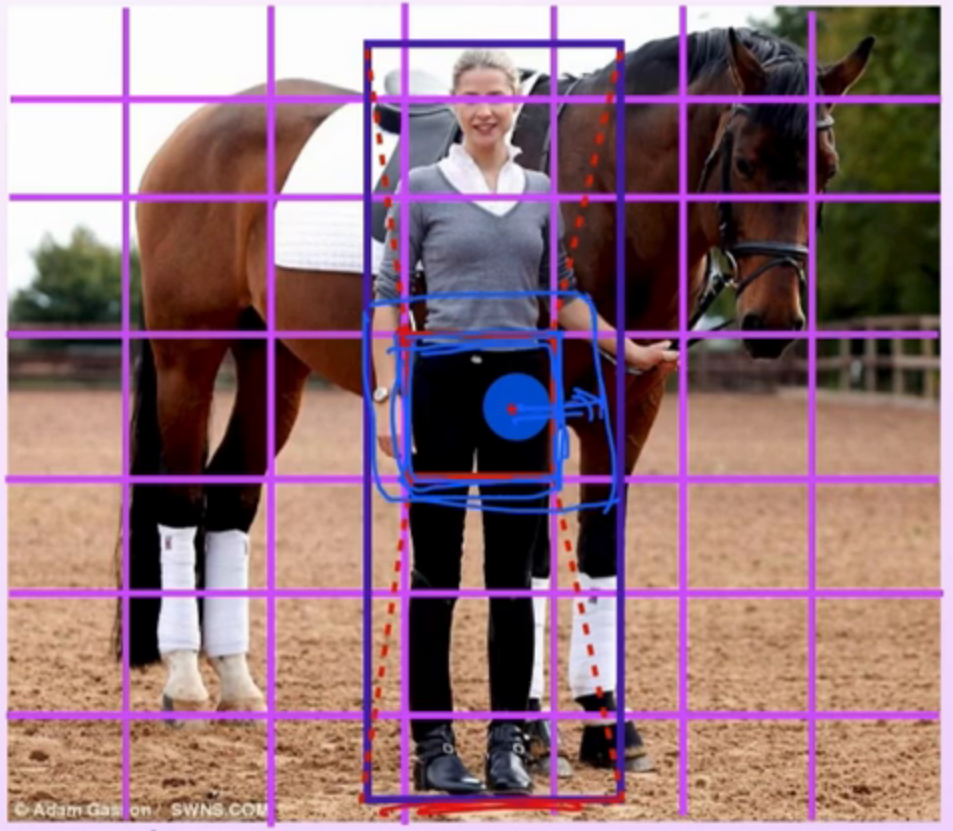
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head





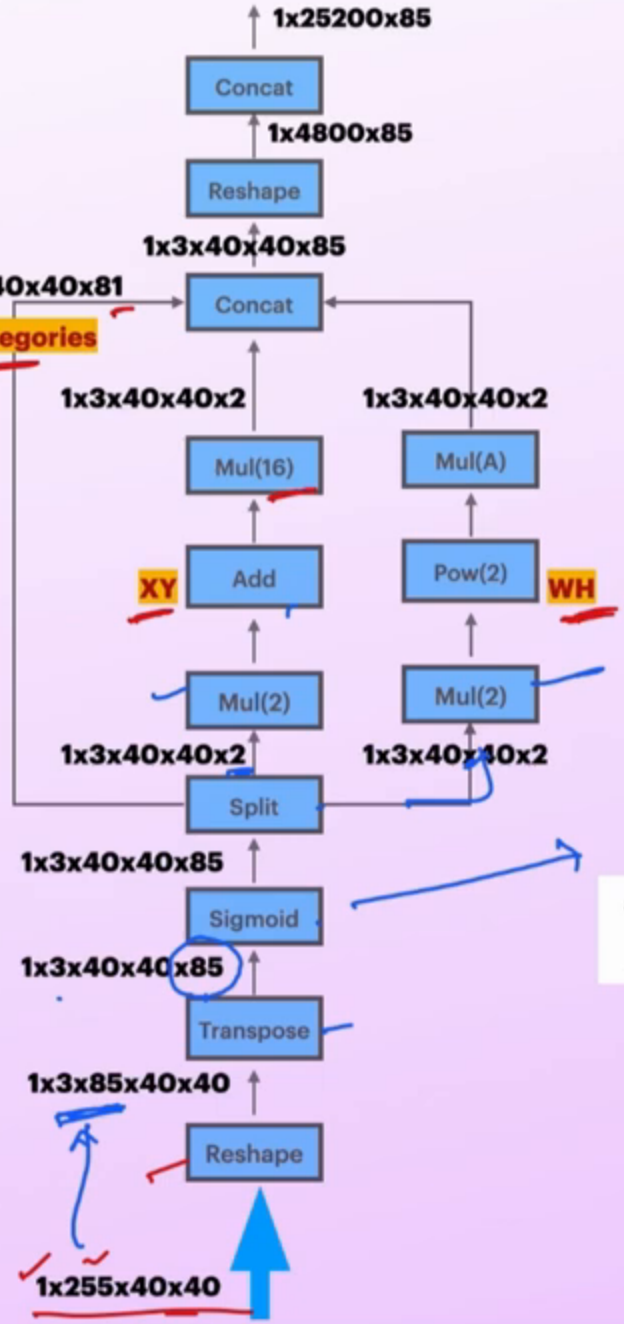


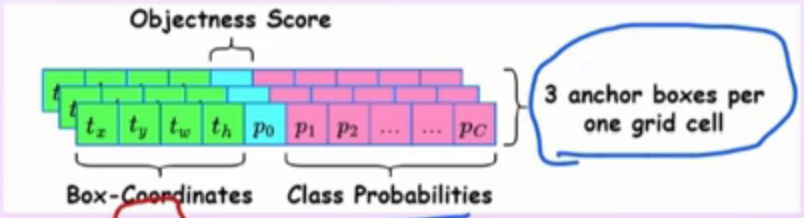
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

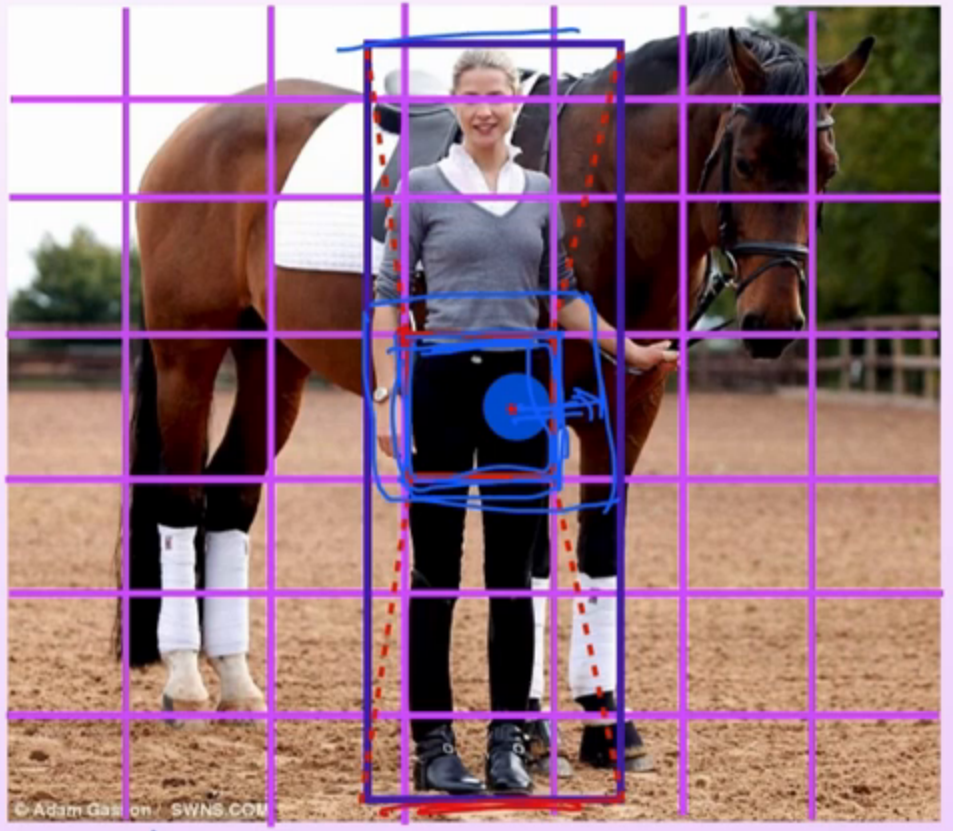
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





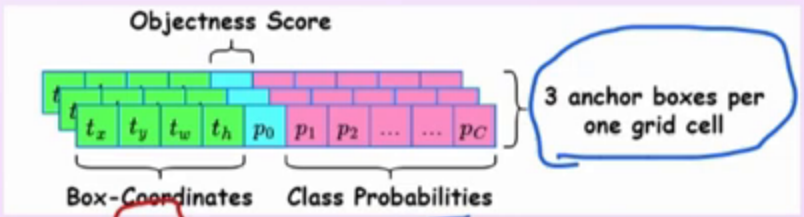


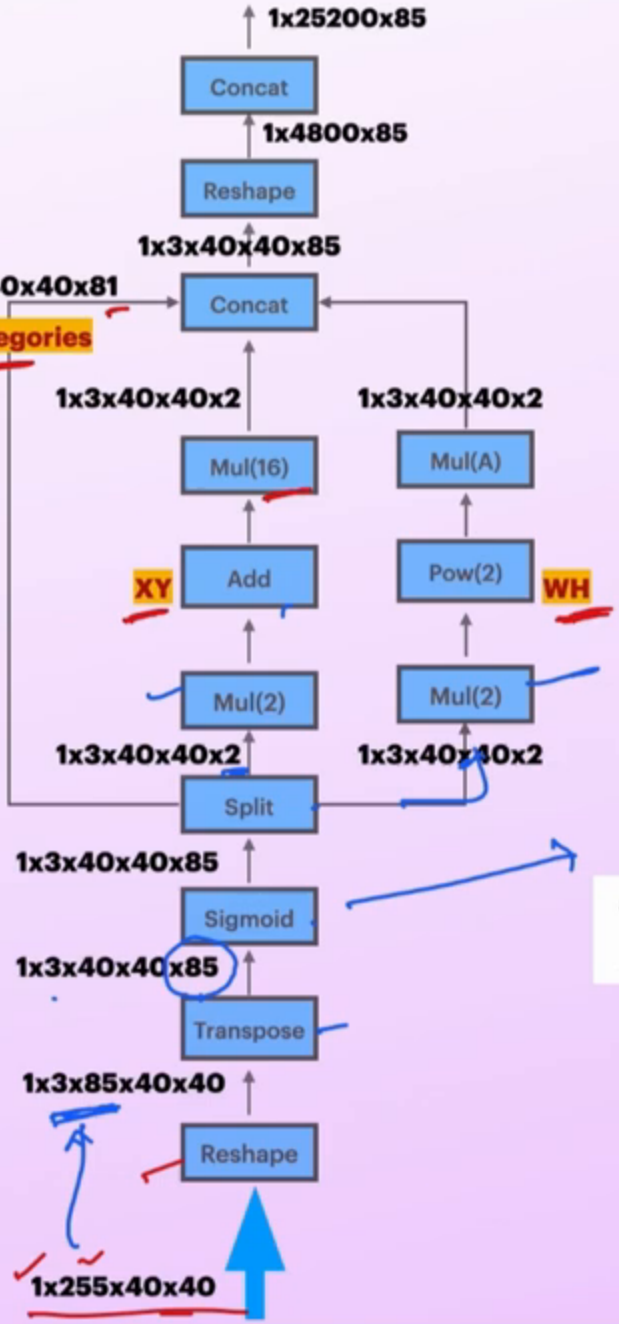
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

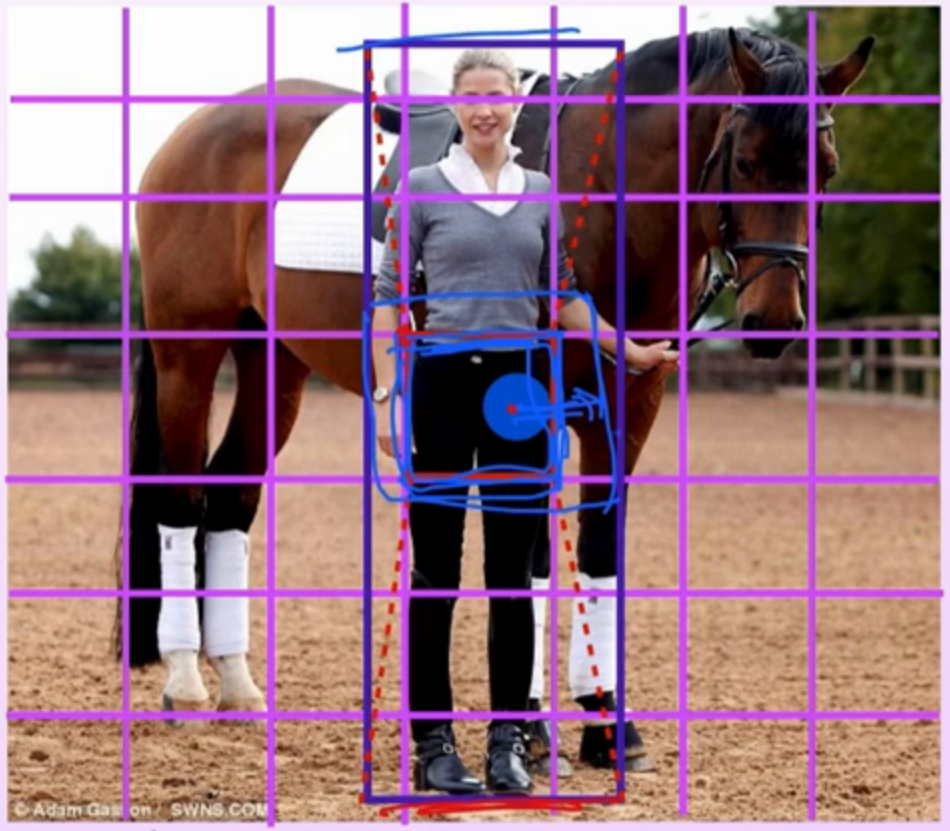
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





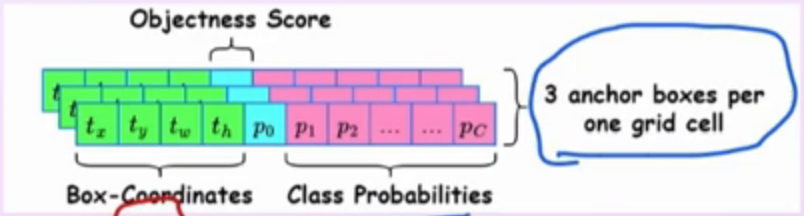


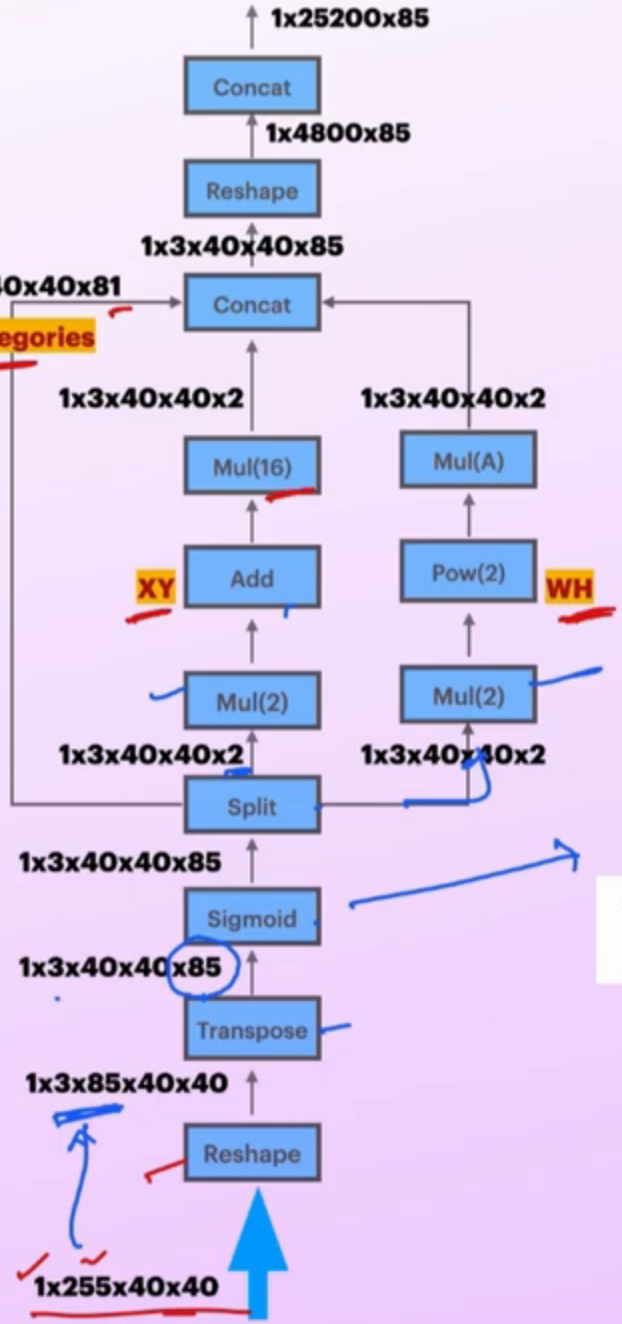
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

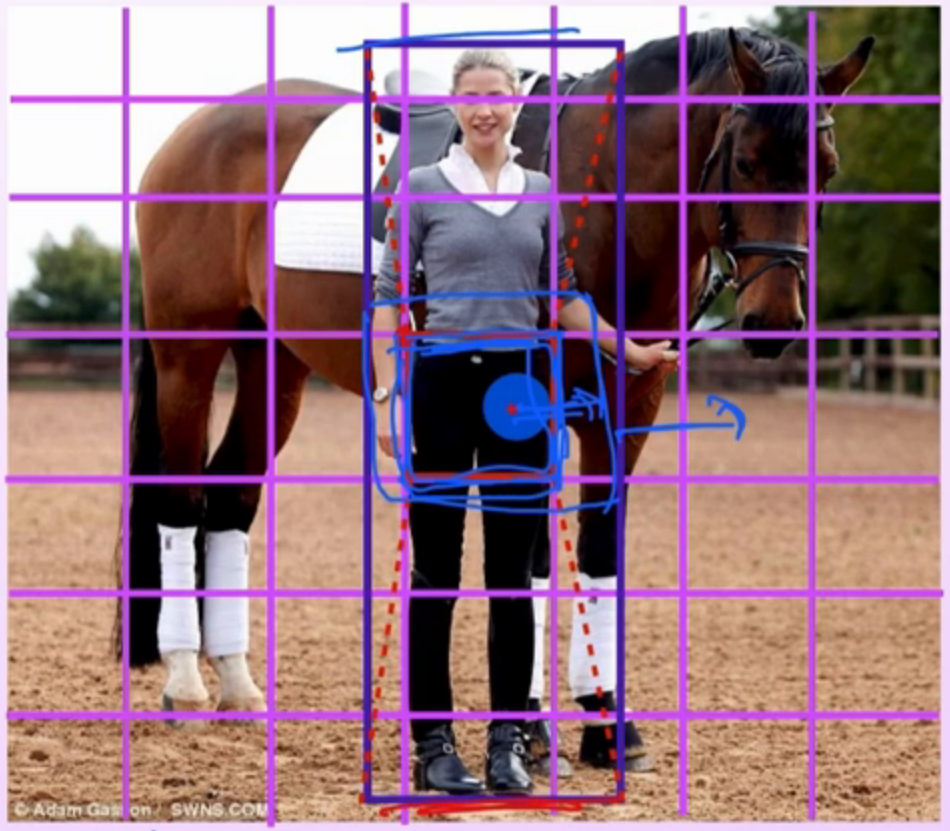
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





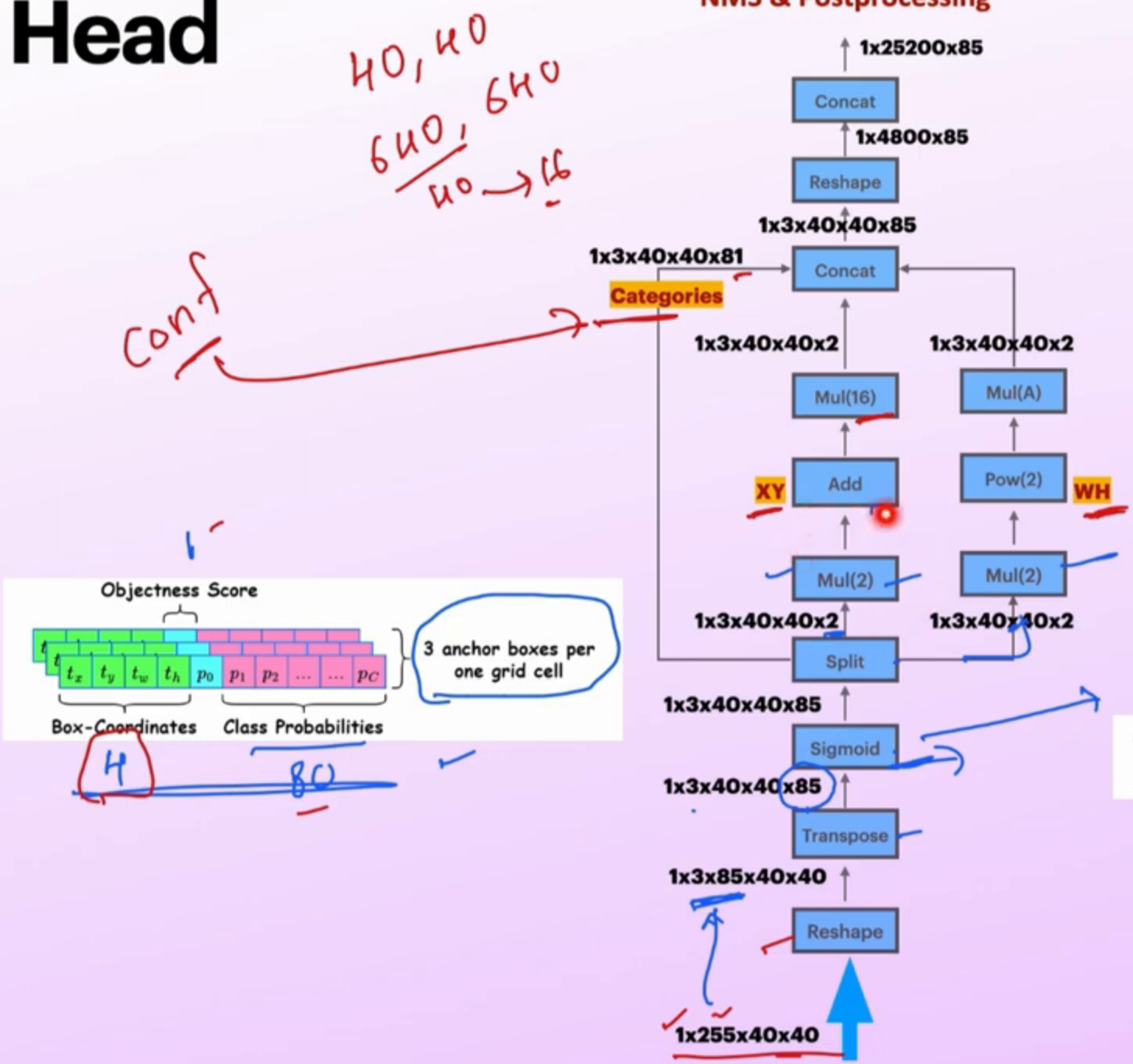


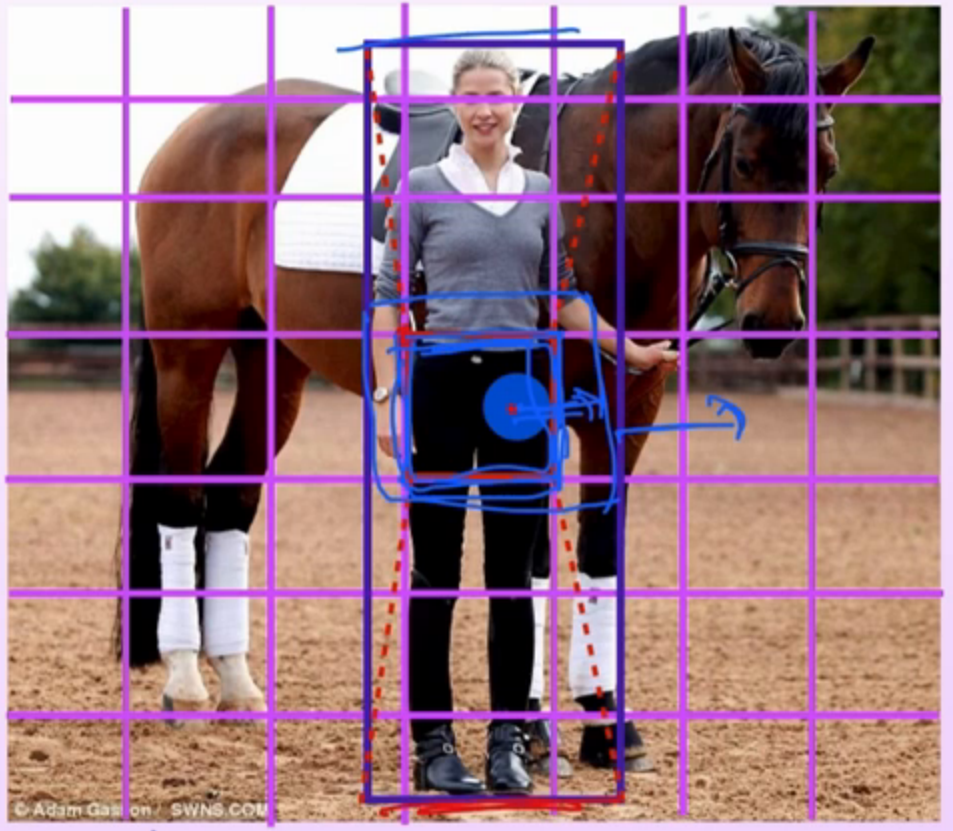
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing



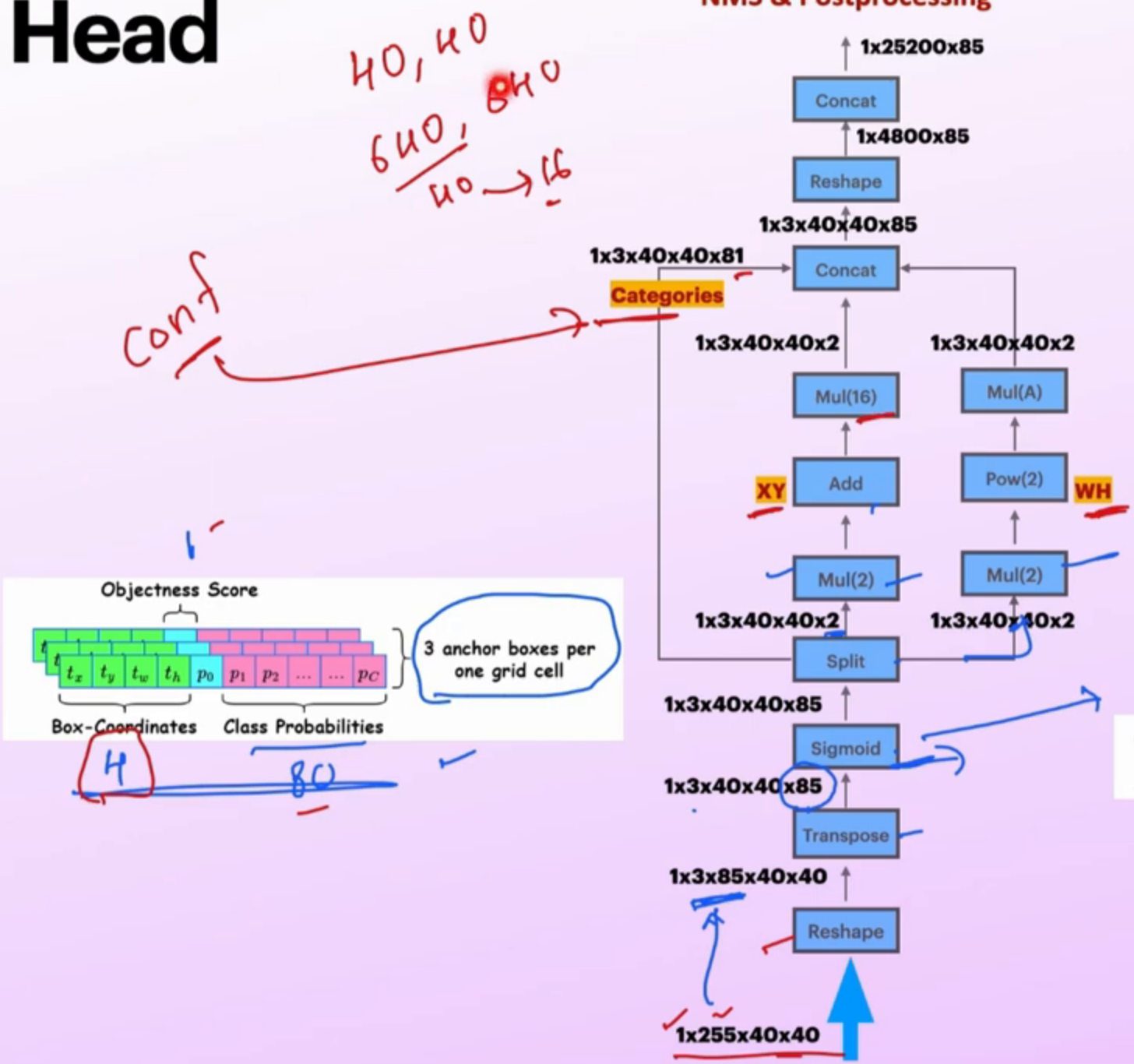


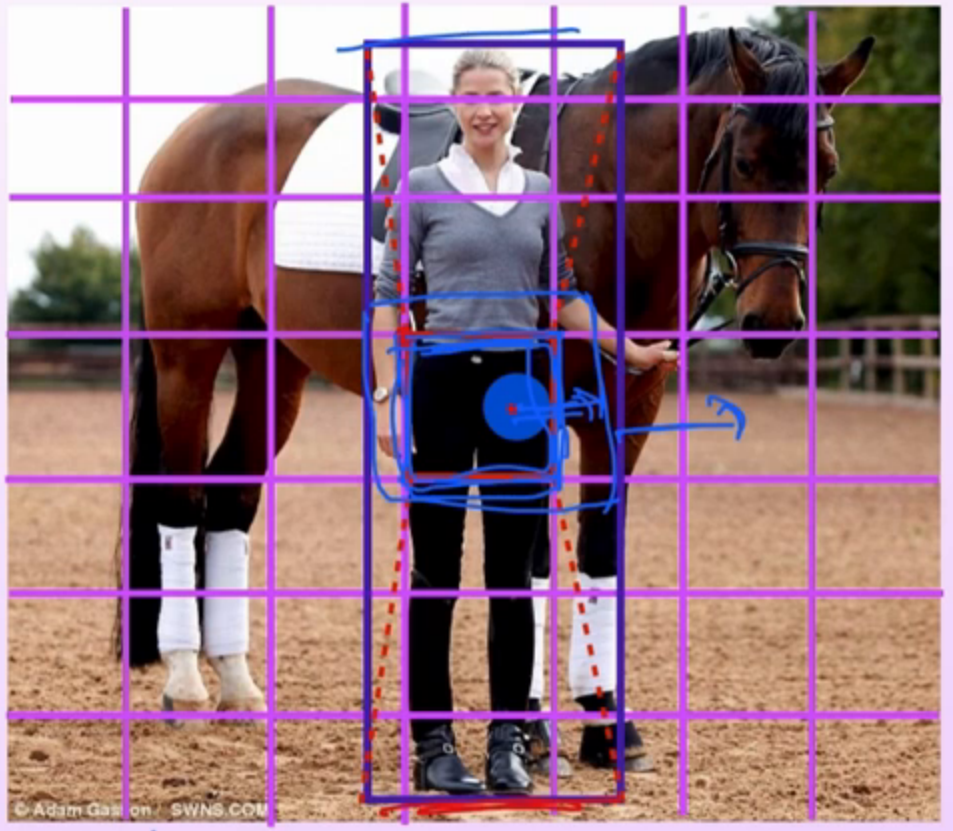
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing



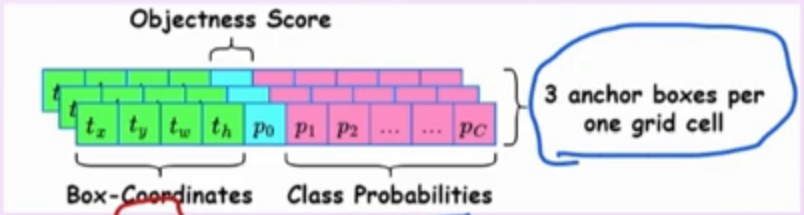


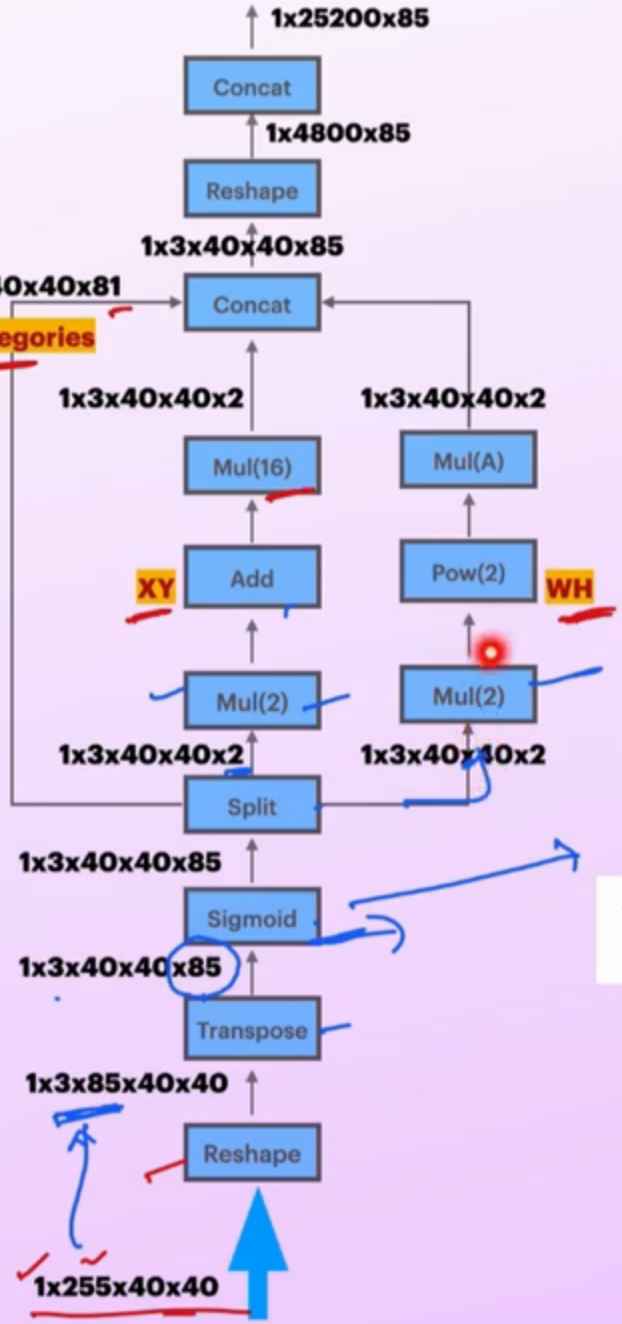
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

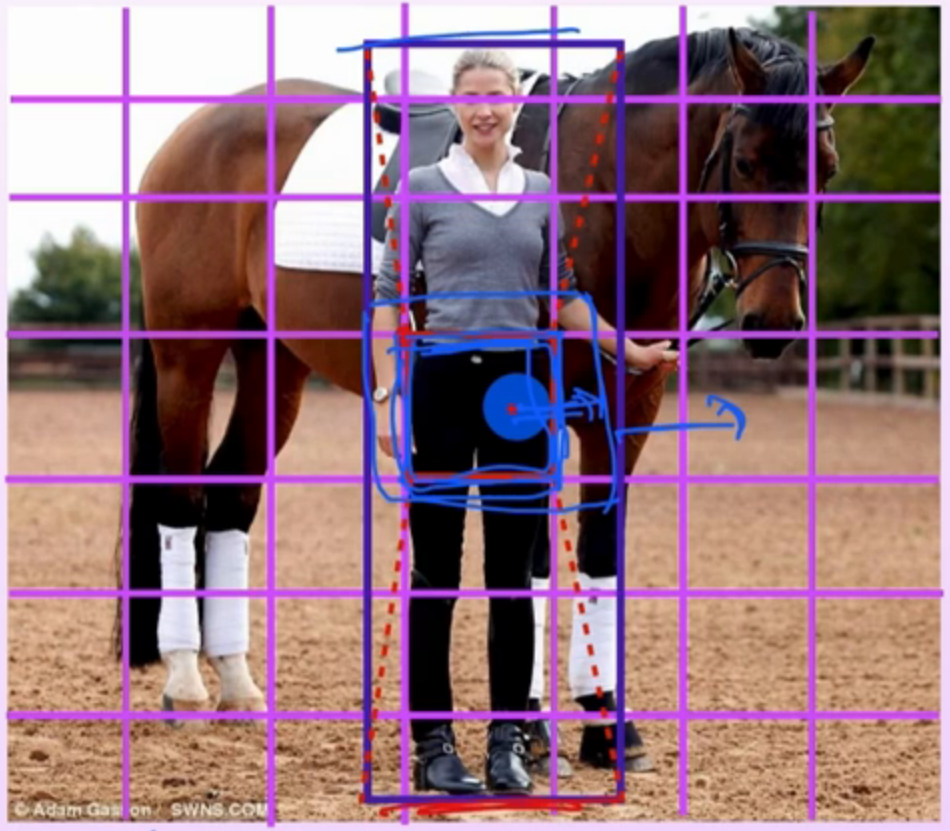
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





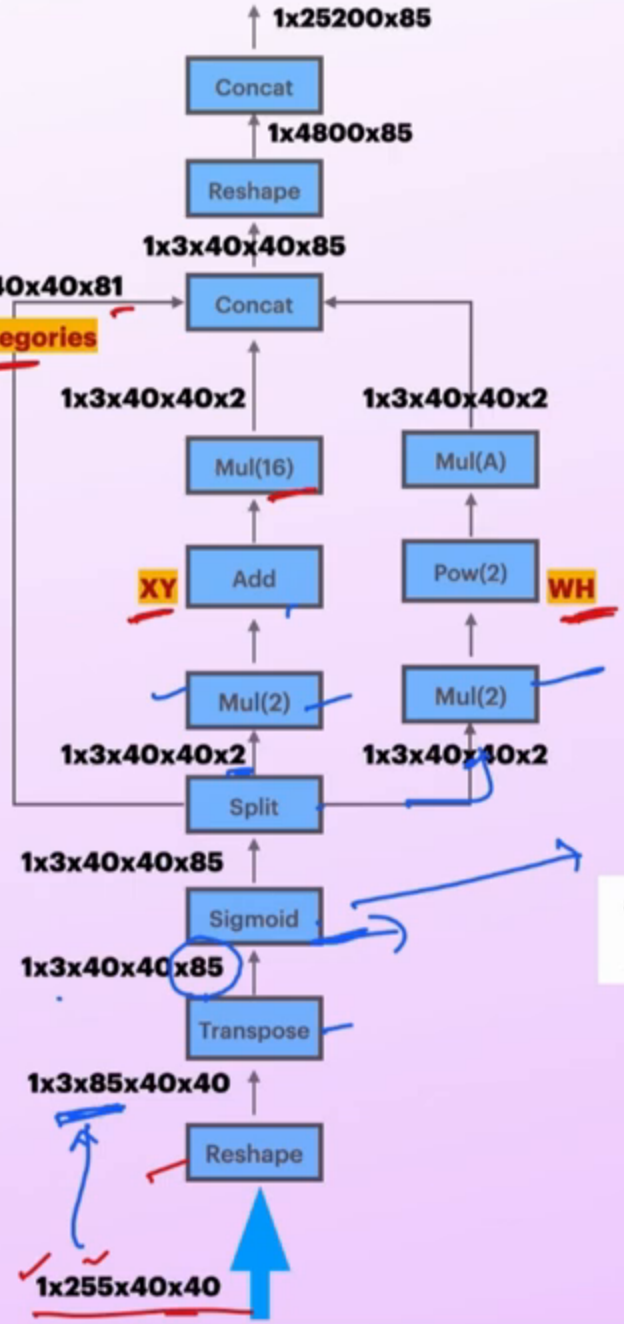


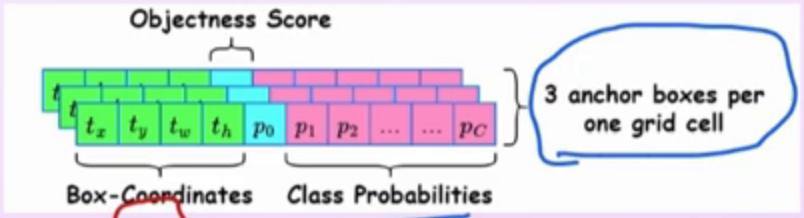
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

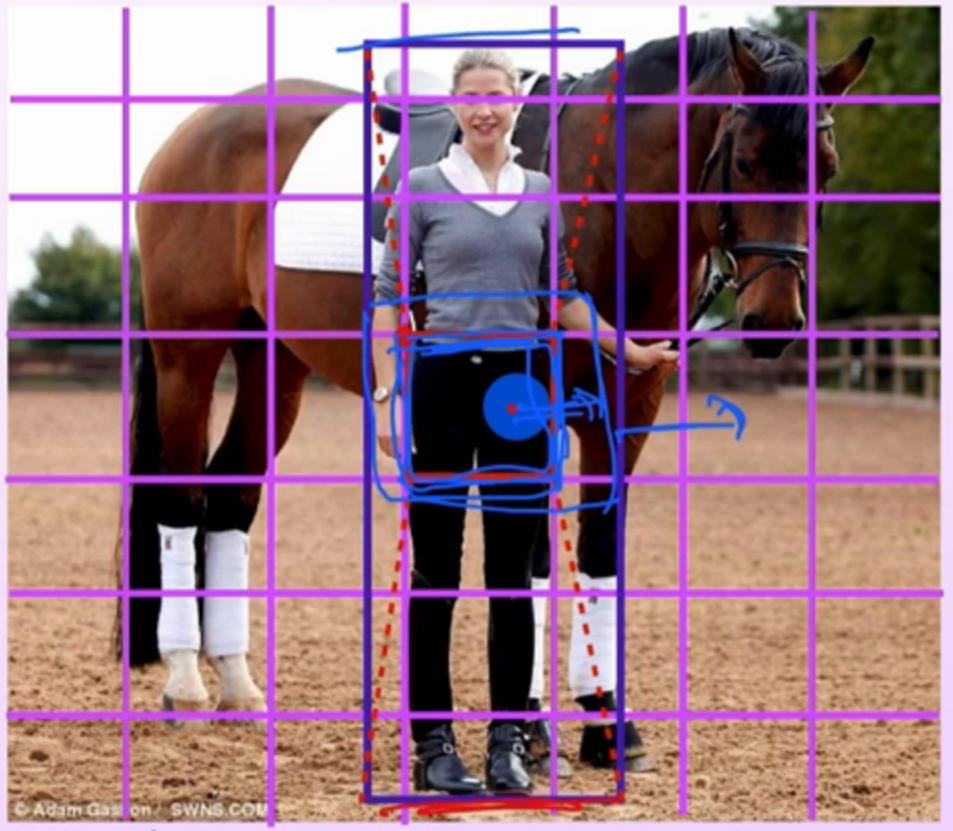
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





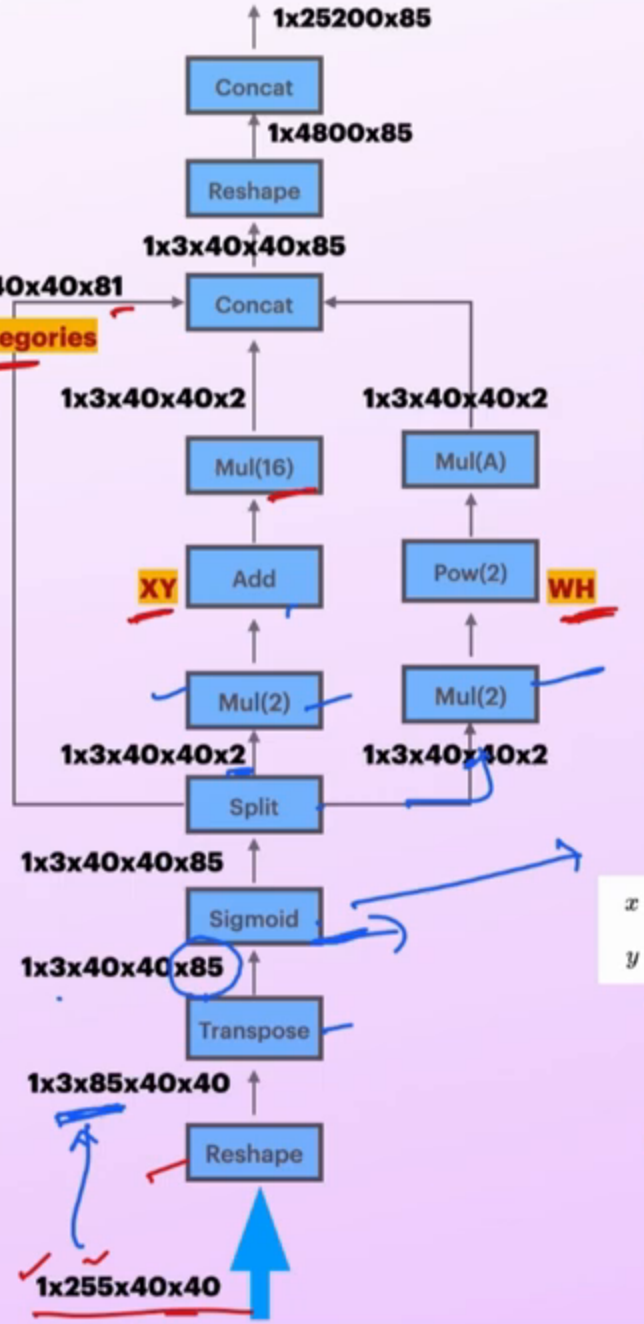


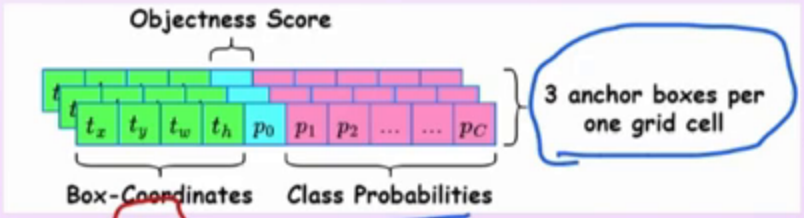
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

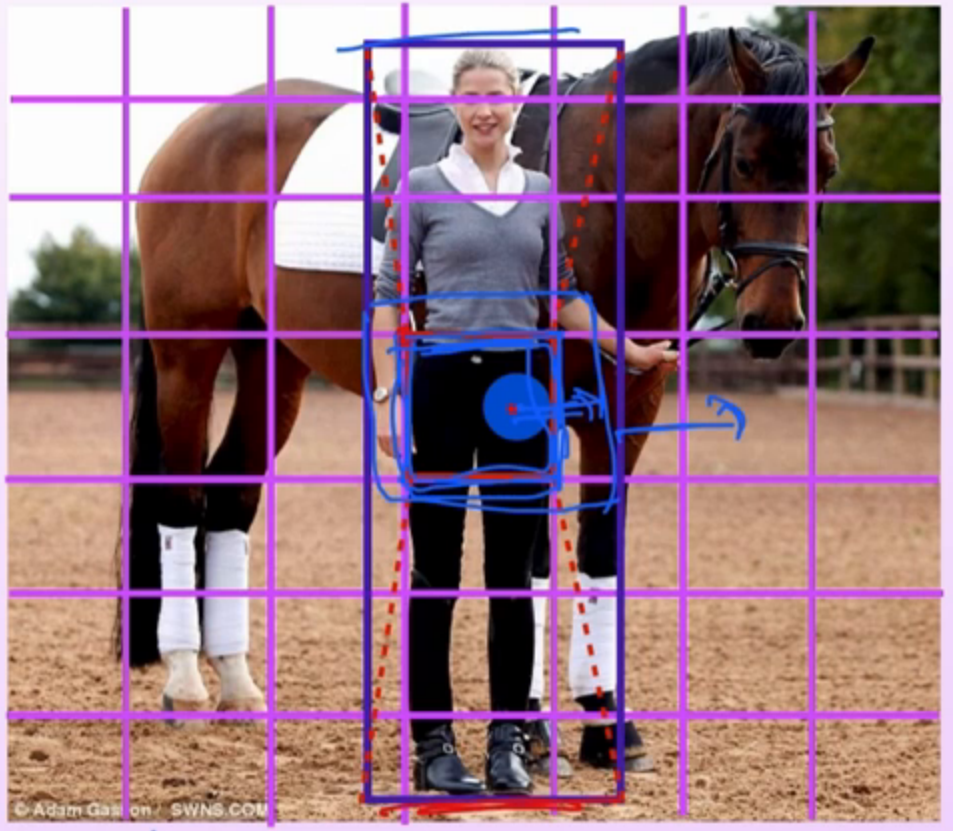
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





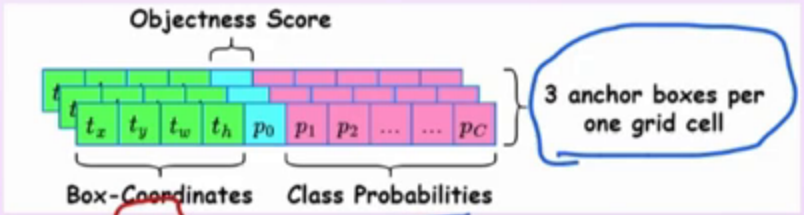


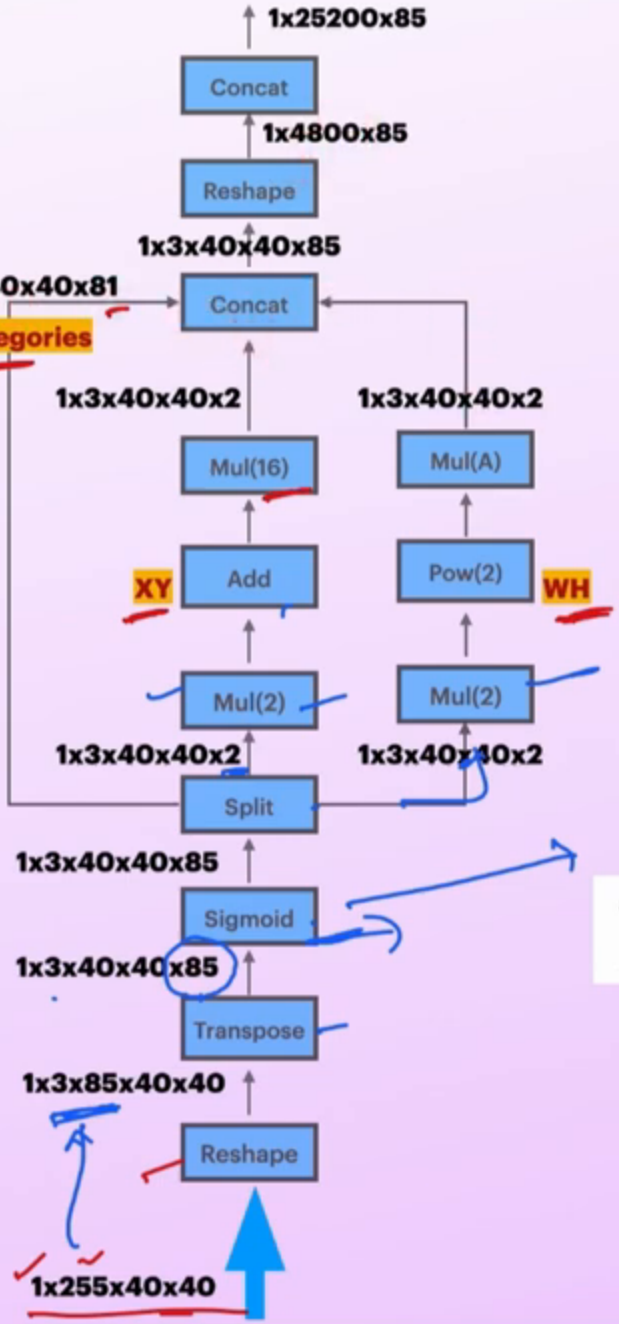
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

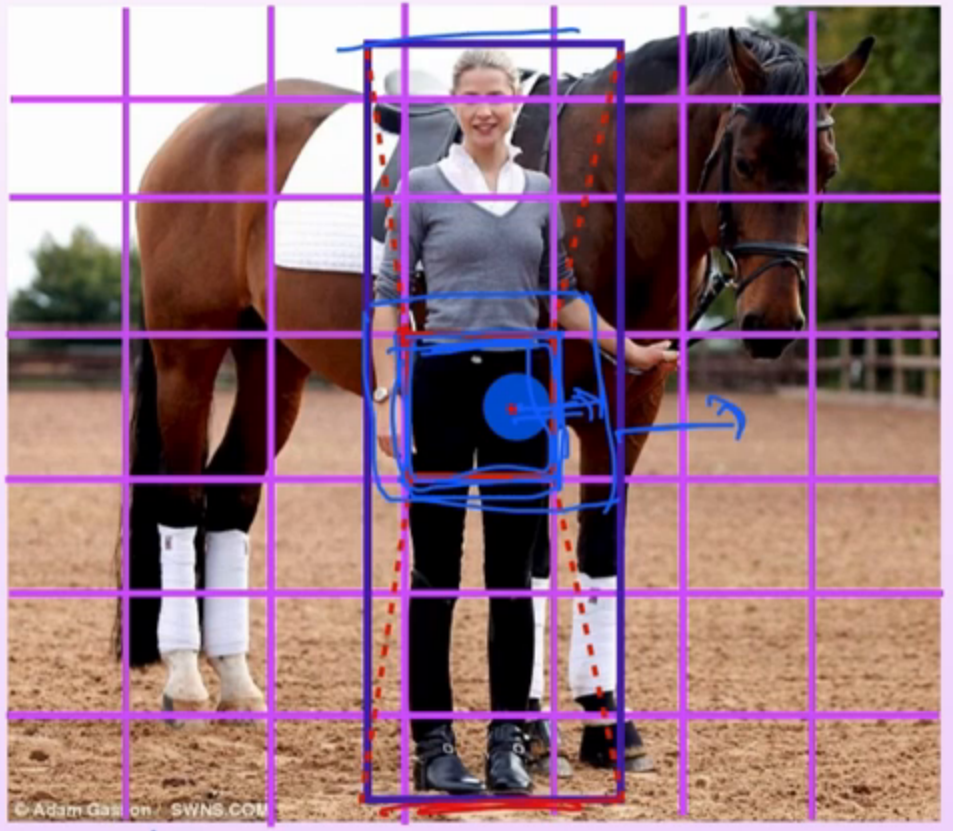
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing





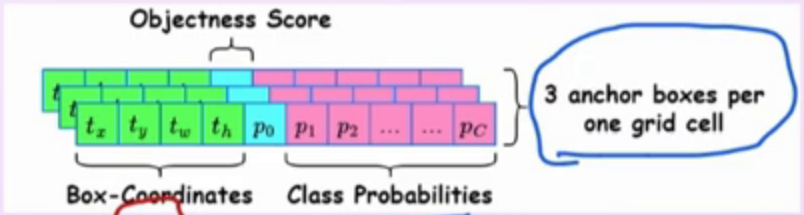


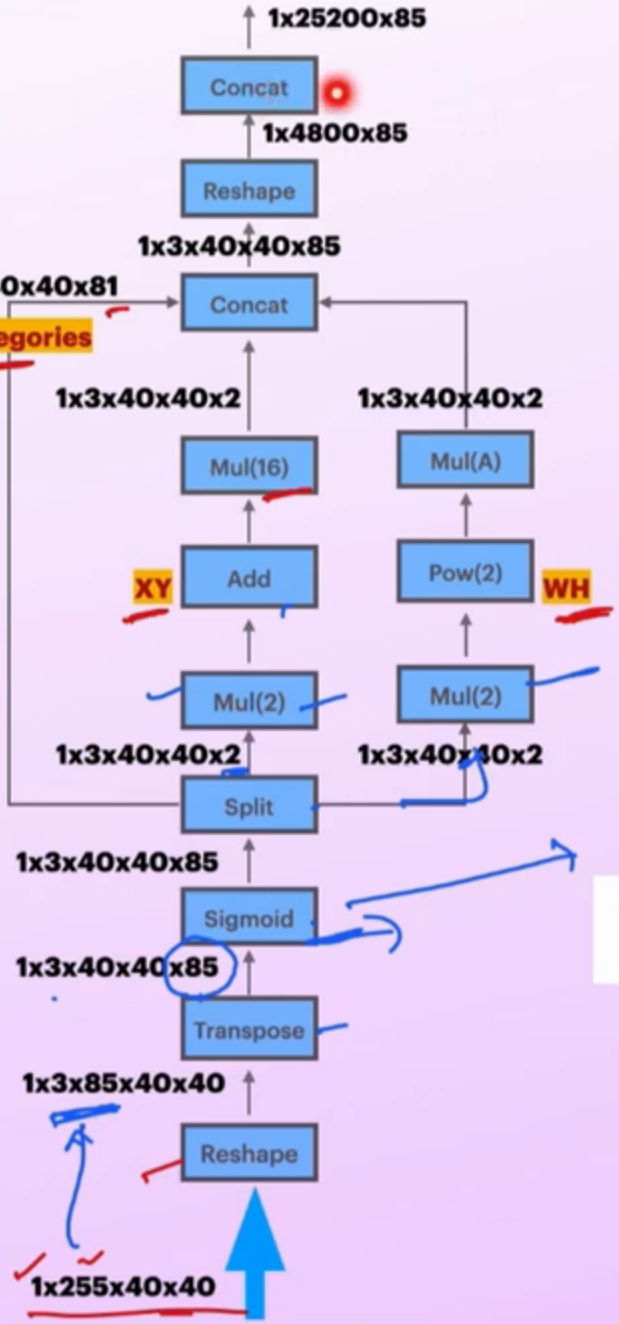
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

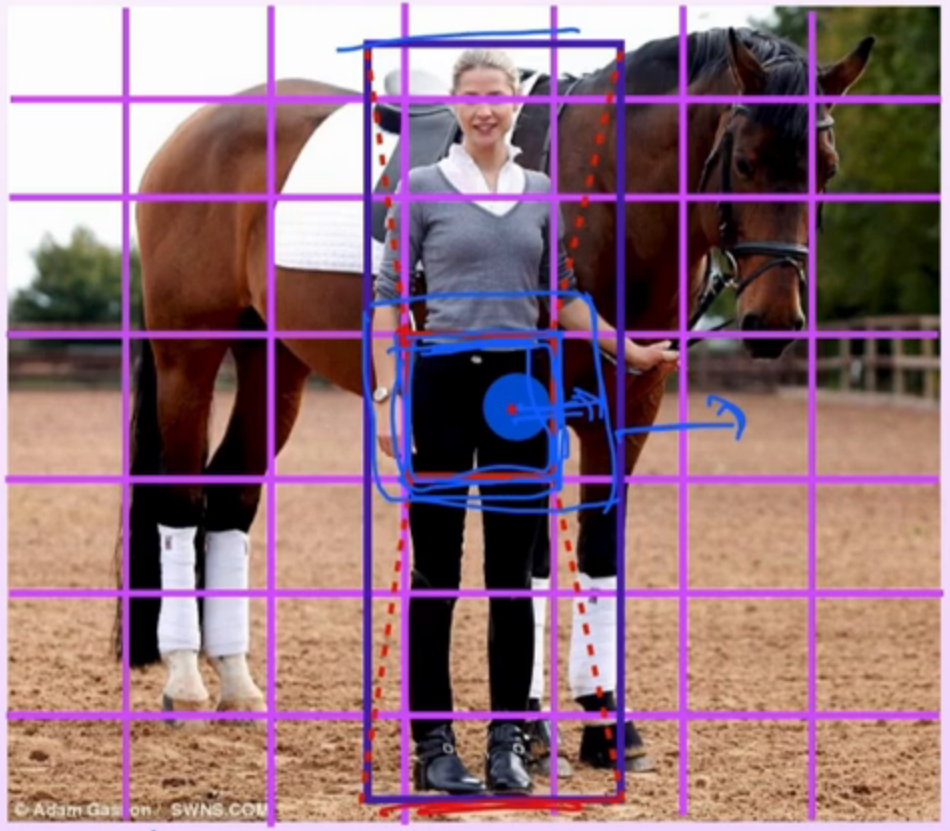
$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

NMS & Postprocessing



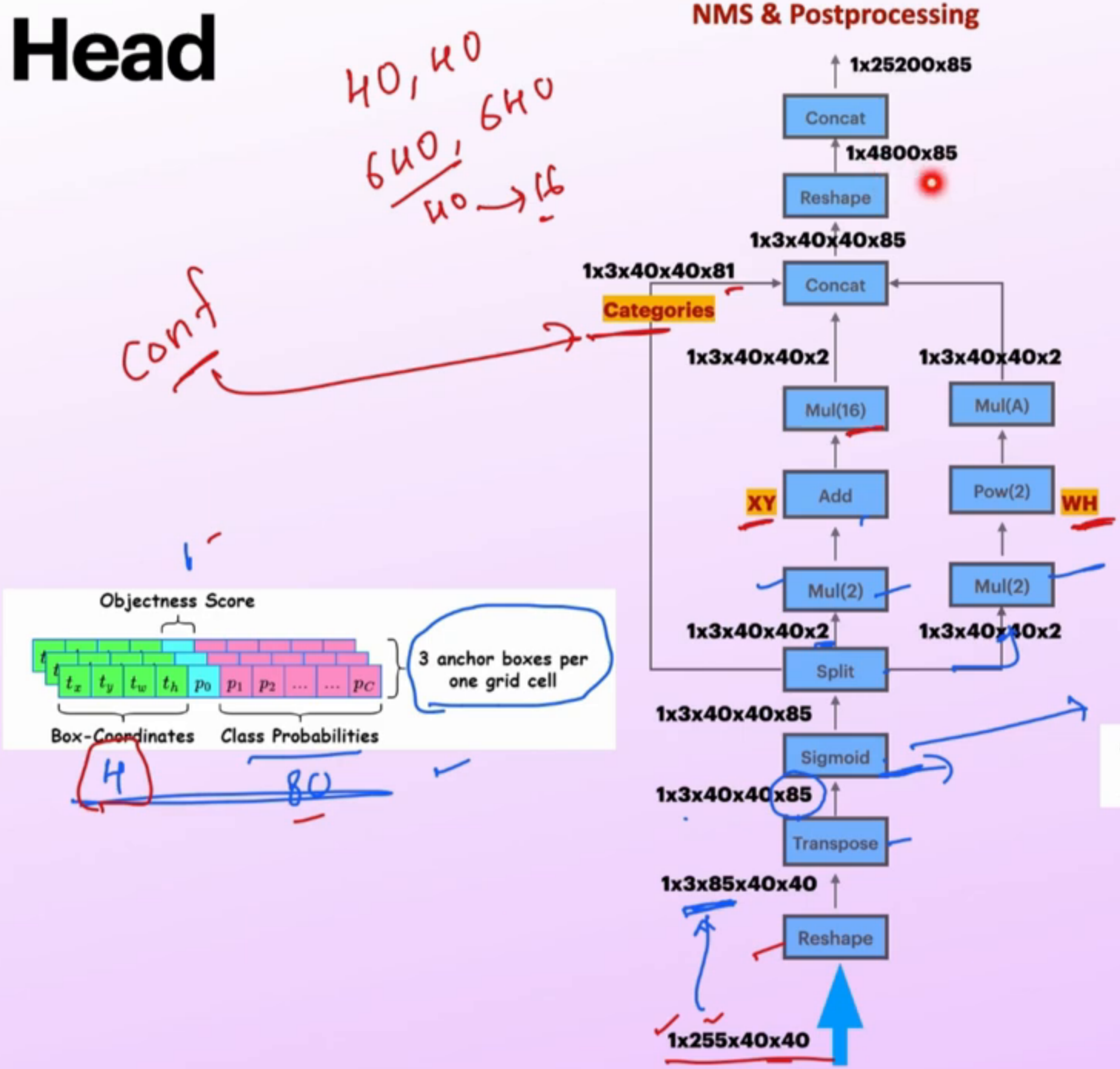


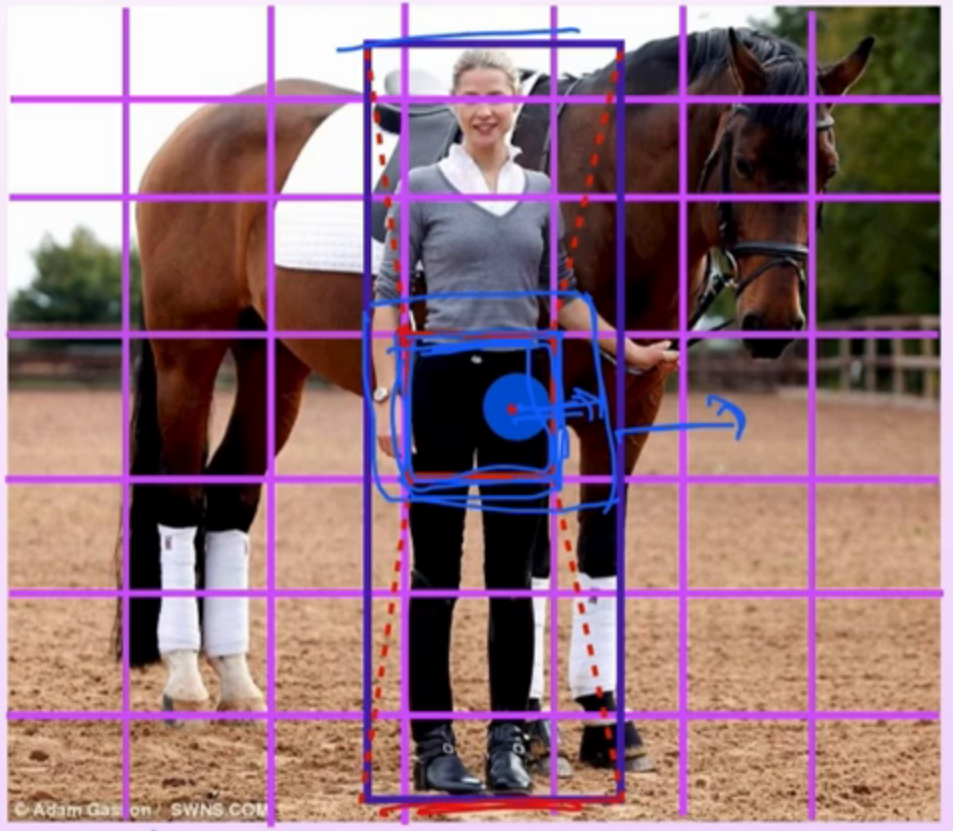


$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

Head

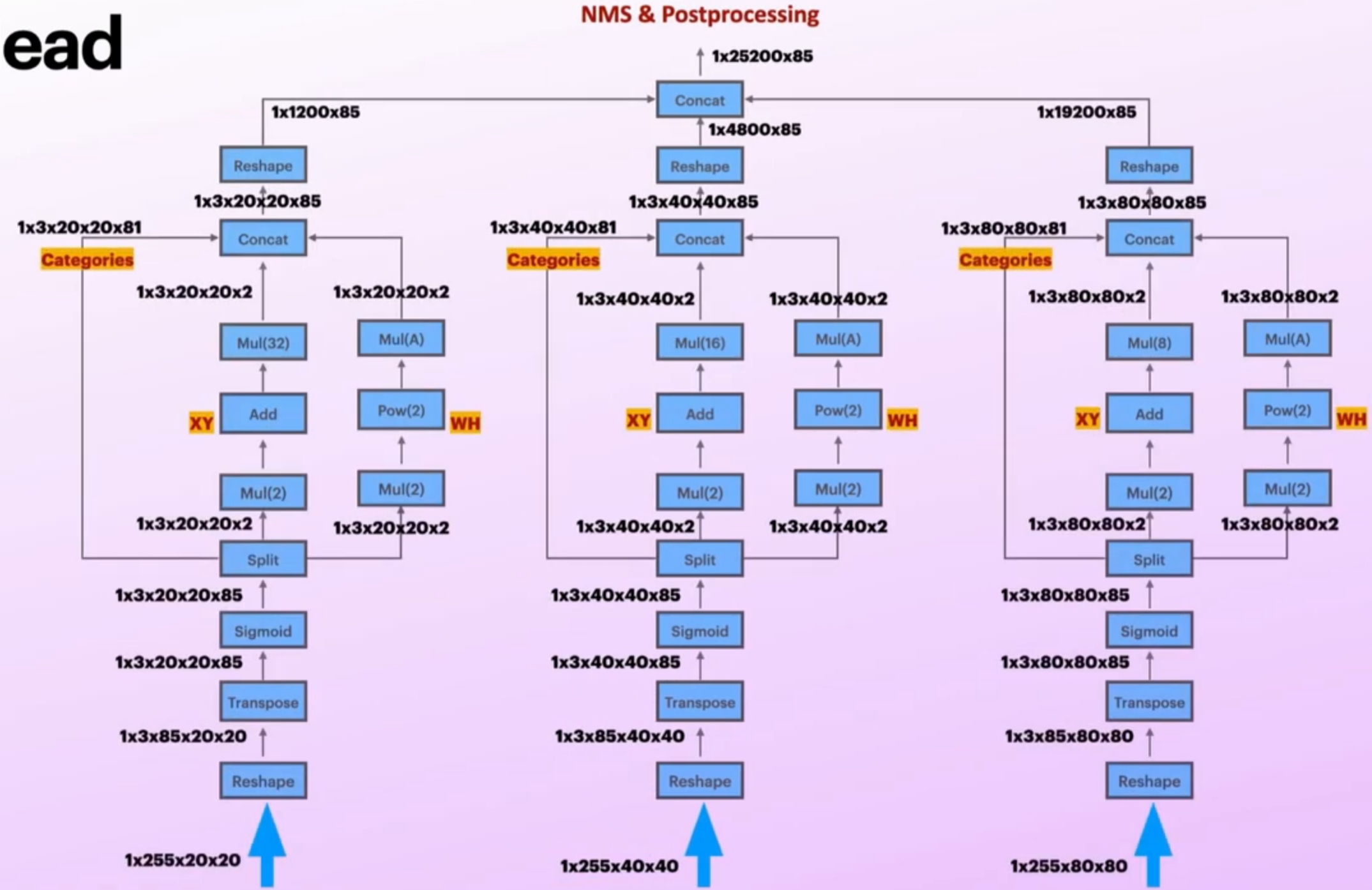




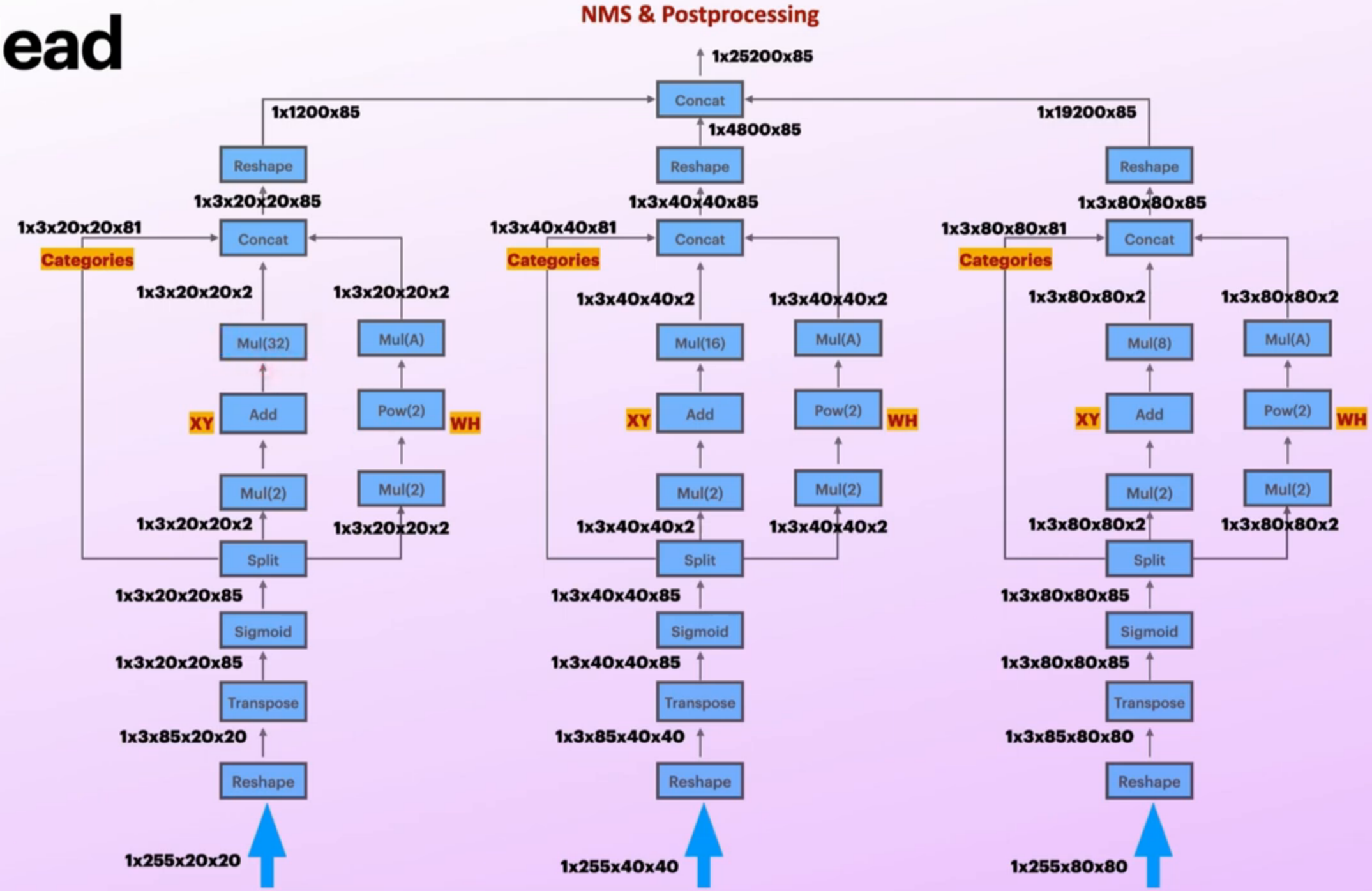
$$x = ((2 \cdot \sigma(t_x)) - 0.5 + c_x) \times stride$$
$$y = ((2 \cdot \sigma(t_y)) - 0.5 + c_y) \times stride$$

$$w = p_w \times (2 \cdot \sigma(t_w))^2$$
$$h = p_h \times (2 \cdot \sigma(t_h))^2$$

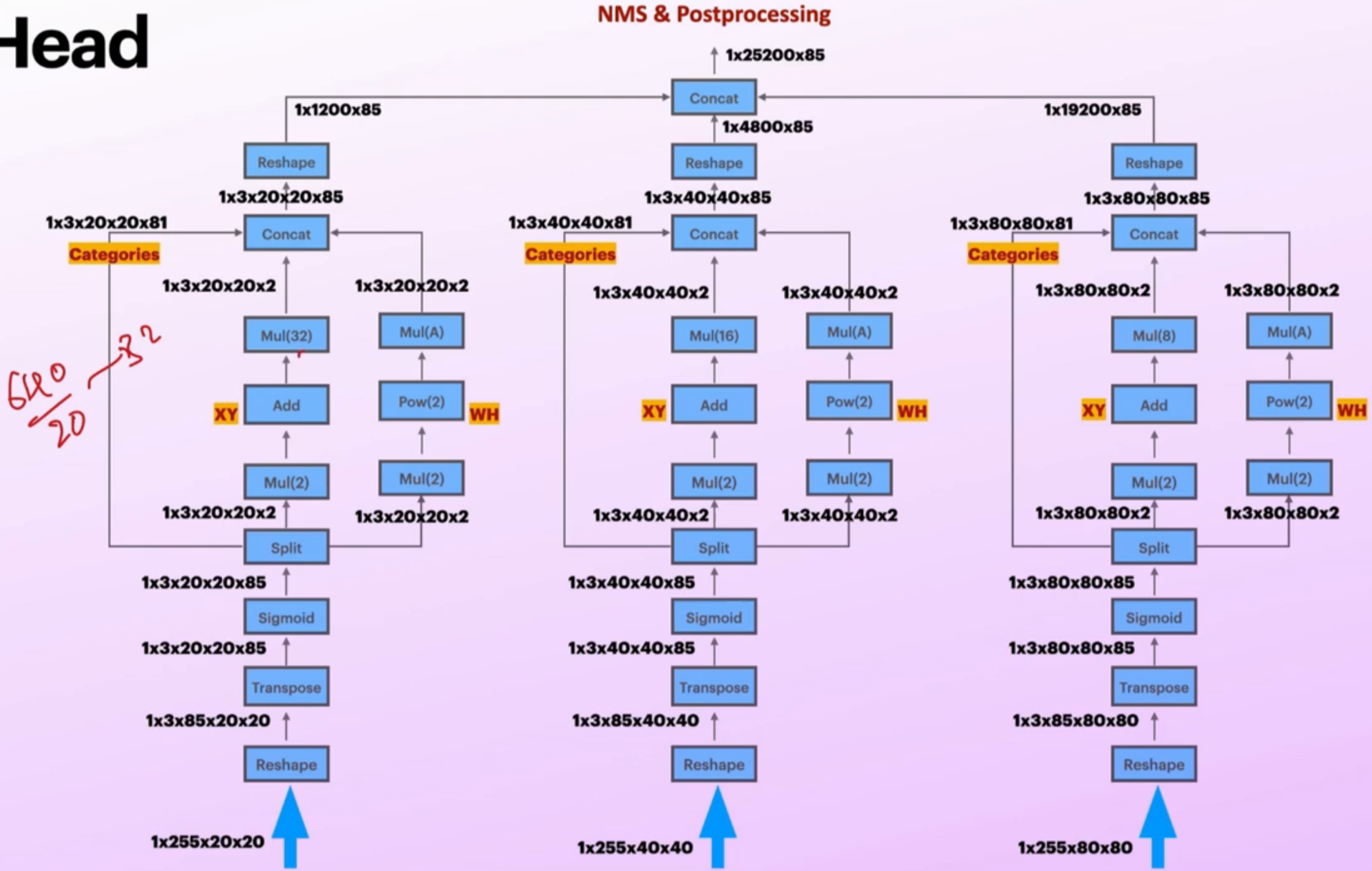
Head


NMS & Postprocessing
1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Reshape
Reshape
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
1x3x20x20x81
1x3x40x40x81
1x3x80x80x81
Concat
Concat
Concat
Categories
Categories
Categories
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Mul(32)
Mul(A)
Mul(16)
Mul(A)
Mul(8)
Mul(A)
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Split
Split
Split
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Sigmoid
Sigmoid
Sigmoid
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Transpose
Transpose
Transpose
1x3x85x20x20
1x3x85x40x40
1x3x85x80x80
Reshape
Reshape
Reshape
1x255x20x20
1x255x40x40
1x255x80x80

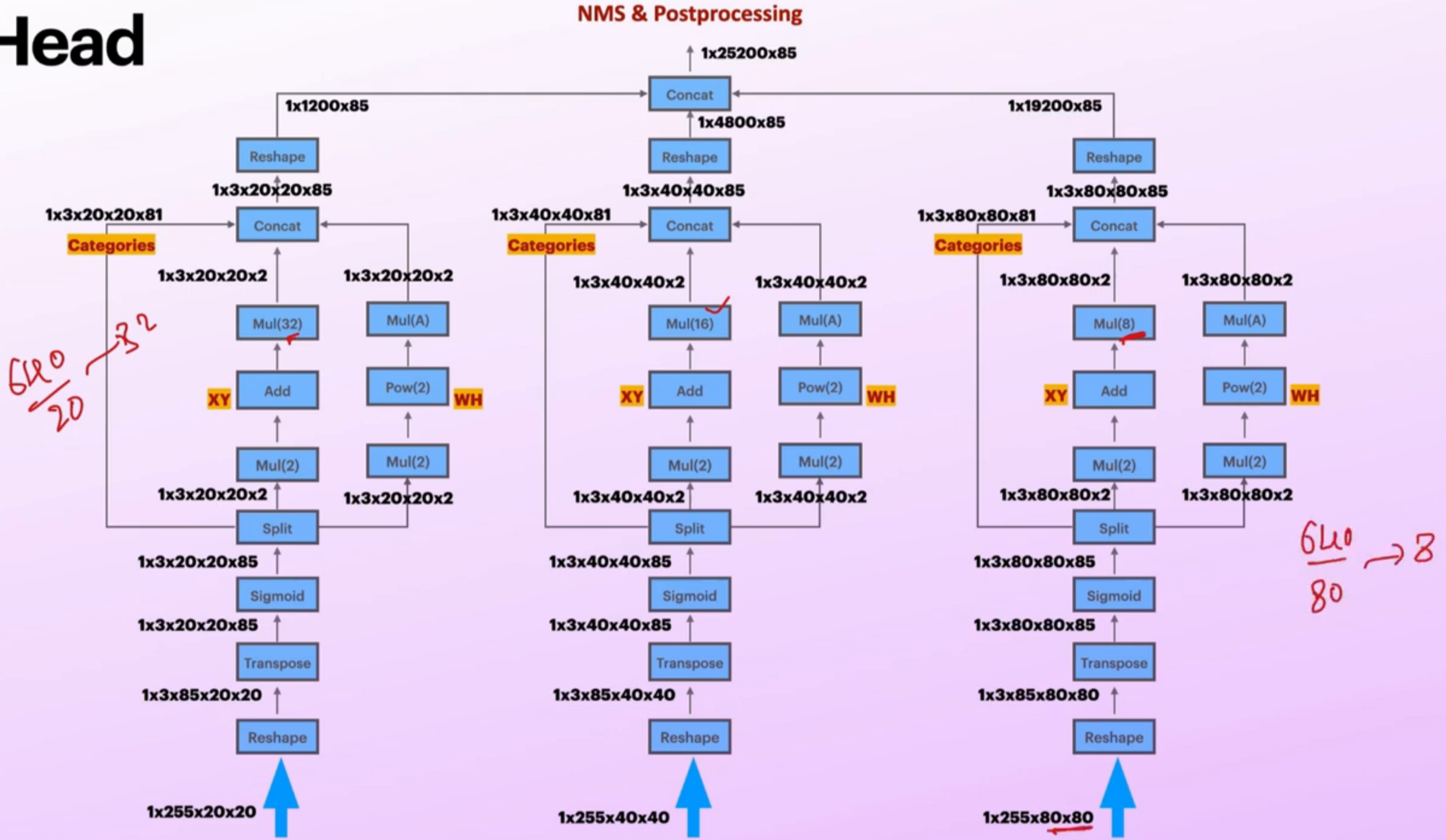
Head


NMS & Postprocessing
1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Reshape
Reshape
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
1x3x20x20x81
1x3x40x40x81
1x3x80x80x81
Concat
Concat
Concat
Categories
Categories
Categories
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Mul(32)
Mul(A)
Mul(16)
Mul(A)
Mul(8)
Mul(A)
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Split
Split
Split
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Sigmoid
Sigmoid
Sigmoid
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Transpose
Transpose
Transpose
1x3x85x20x20
1x3x85x40x40
1x3x85x80x80
Reshape
Reshape
Reshape
1x255x20x20
1x255x40x40
1x255x80x80

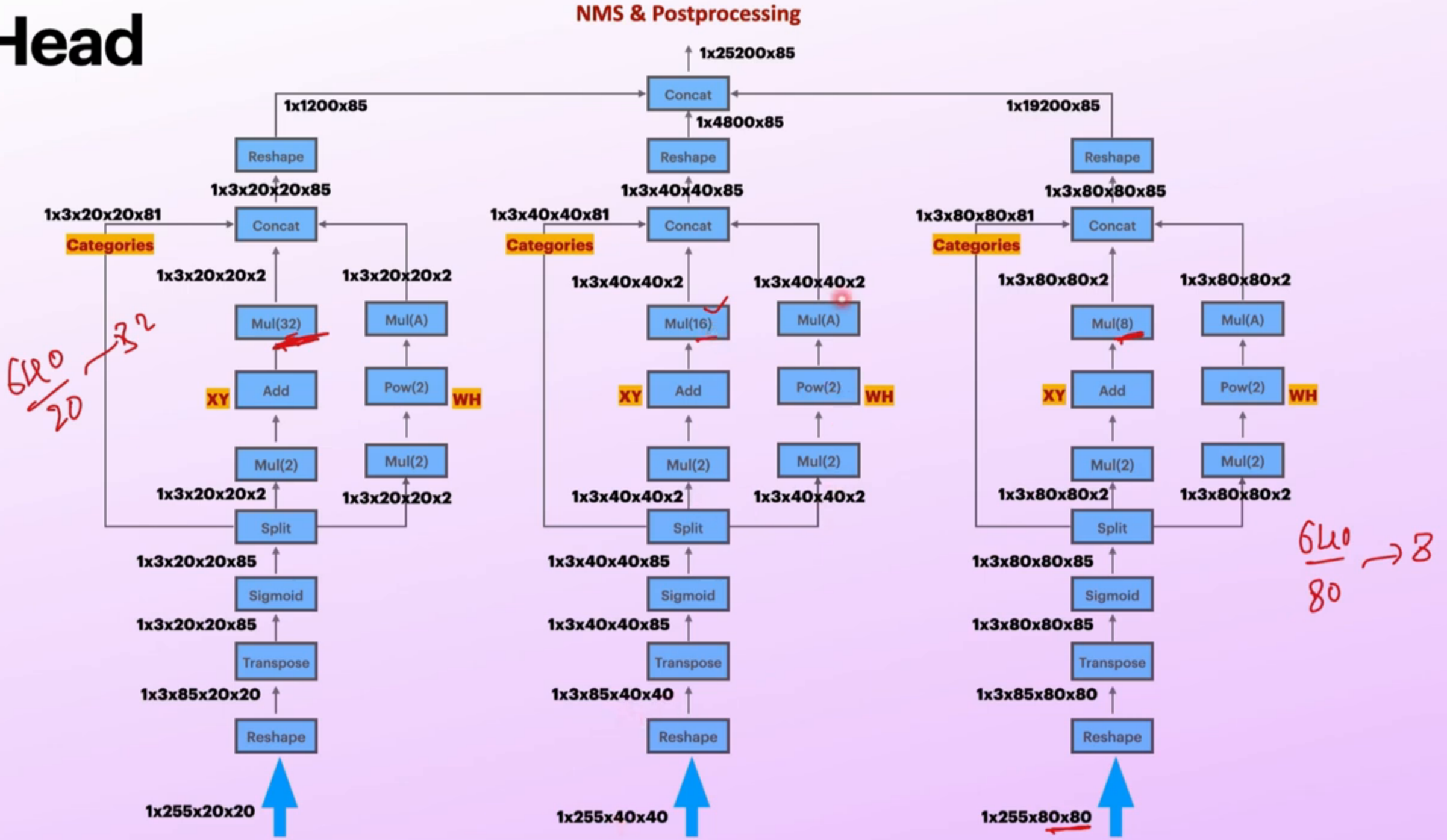
Head


NMS & Postprocessing
1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Reshape
Reshape
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
1x3x20x20x81
1x3x40x40x81
1x3x80x80x81
Concat
Concat
Concat
Categories
Categories
Categories
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Mul(32)
Mul(A)
Mul(16)
Mul(A)
Mul(8)
Mul(A)
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Split
Split
Split
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Sigmoid
Sigmoid
Sigmoid
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Transpose
Transpose
Transpose
1x3x85x20x20
1x3x85x40x40
1x3x85x80x80
Reshape
Reshape
Reshape
1x255x20x20
1x255x40x40
1x255x80x80

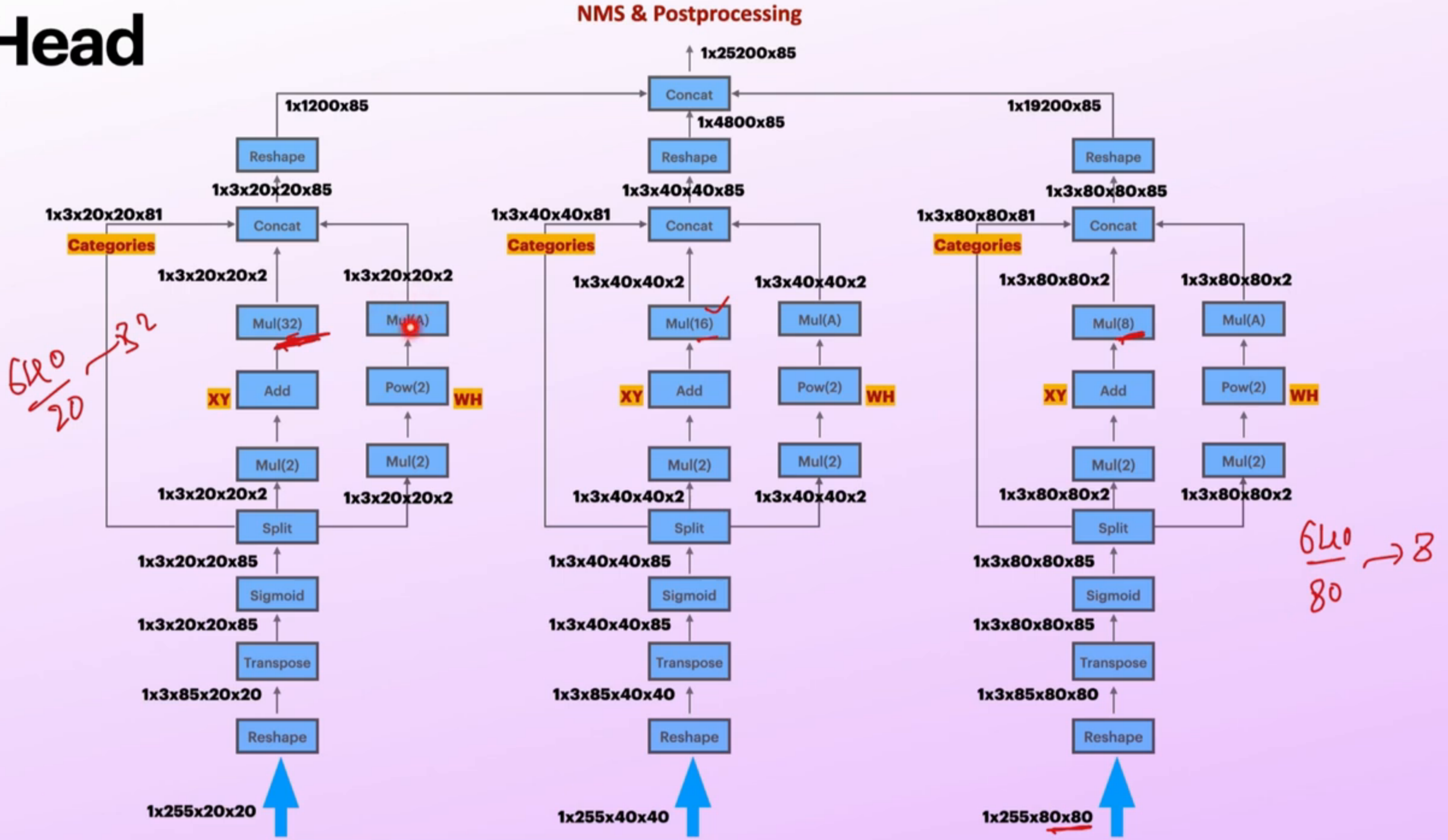
Head


NMS & Postprocessing
1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Reshape
Reshape
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
1x3x20x20x81
1x3x40x40x81
1x3x80x80x81
Concat
Concat
Concat
Categories
Categories
Categories
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Mul(32)
Mul(A)
Mul(16)
Mul(A)
Mul(8)
Mul(A)
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Split
Split
Split
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Sigmoid
Sigmoid
Sigmoid
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Transpose
Transpose
Transpose
1x3x85x20x20
1x3x85x40x40
1x3x85x80x80
Reshape
Reshape
Reshape
1x255x20x20
1x255x40x40
1x255x80x80
640/20 → 32
640/80 → 8

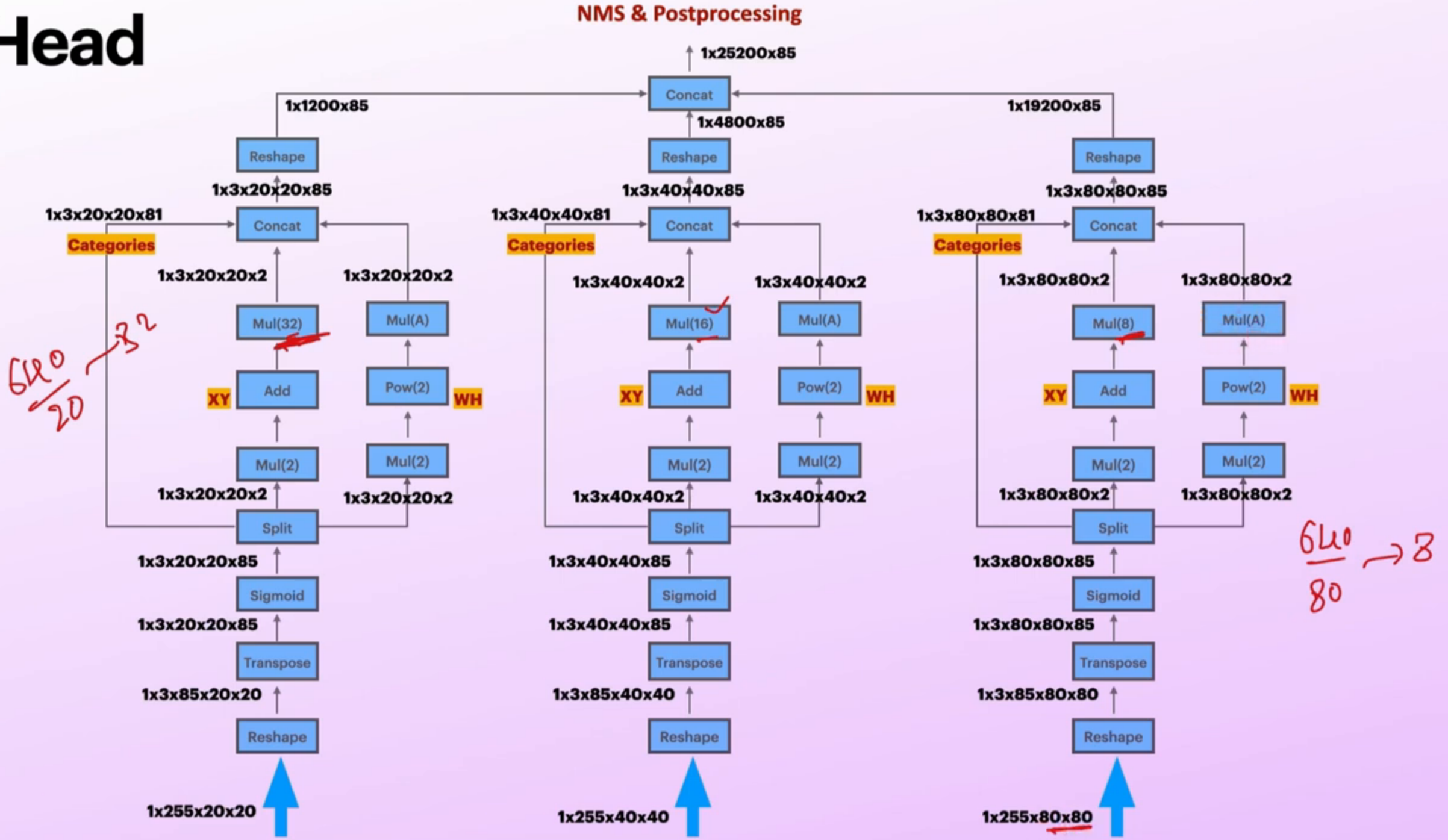
Head


NMS & Postprocessing
1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Reshape
Reshape
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
1x3x20x20x81
1x3x40x40x81
1x3x80x80x81
Concat
Concat
Concat
Categories
Categories
Categories
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Mul(32)
Mul(A)
Mul(16)
Mul(A)
Mul(8)
Mul(A)
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Split
Split
Split
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Sigmoid
Sigmoid
Sigmoid
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Transpose
Transpose
Transpose
1x3x85x20x20
1x3x85x40x40
1x3x85x80x80
Reshape
Reshape
Reshape
1x255x20x20
1x255x40x40
1x255x80x80

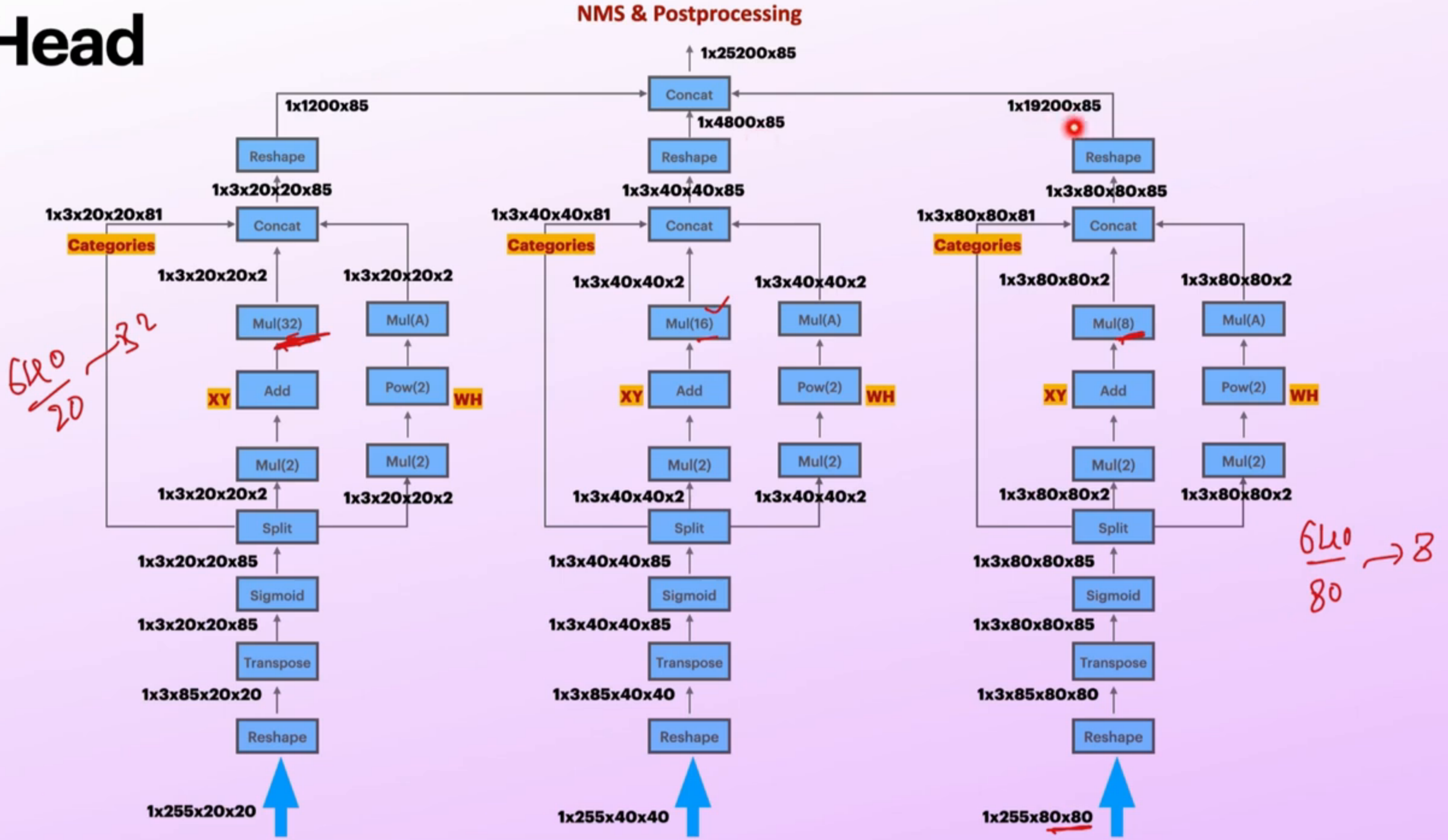
Head


NMS & Postprocessing
1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
1x3x20x20x81
1x3x40x40x81
1x3x80x80x81
Concat
Categories
1x3x20x20x2
1x3x40x40x2
1x3x80x80x2
Mul(32)
Mul(16)
Mul(8)
Mul(A)
XY
Add
Pow(2)
WH
Mul(2)
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Split
Sigmoid
1x3x85x20x20
1x3x85x40x40
1x3x85x80x80
Transpose
Reshape
1x255x20x20
1x255x40x40
1x255x80x80

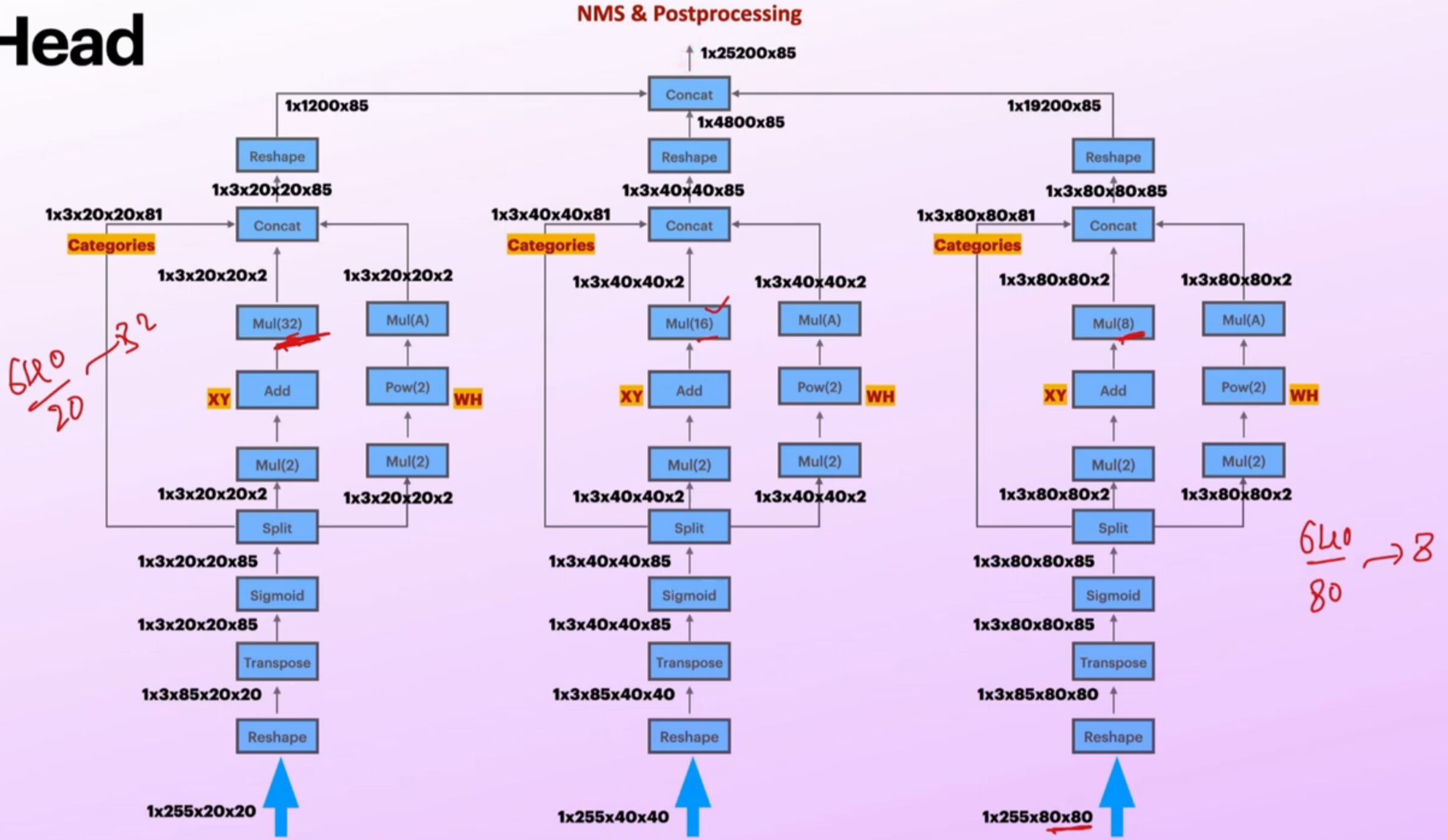
Head


NMS & Postprocessing
1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Reshape
Reshape
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
1x3x20x20x81
1x3x40x40x81
1x3x80x80x81
Concat
Concat
Concat
Categories
Categories
Categories
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Mul(32)
Mul(A)
Mul(16)
Mul(A)
Mul(8)
Mul(A)
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Split
Split
Split
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Sigmoid
Sigmoid
Sigmoid
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Transpose
Transpose
Transpose
1x3x85x20x20
1x3x85x40x40
1x3x85x80x80
Reshape
Reshape
Reshape
1x255x20x20
1x255x40x40
1x255x80x80

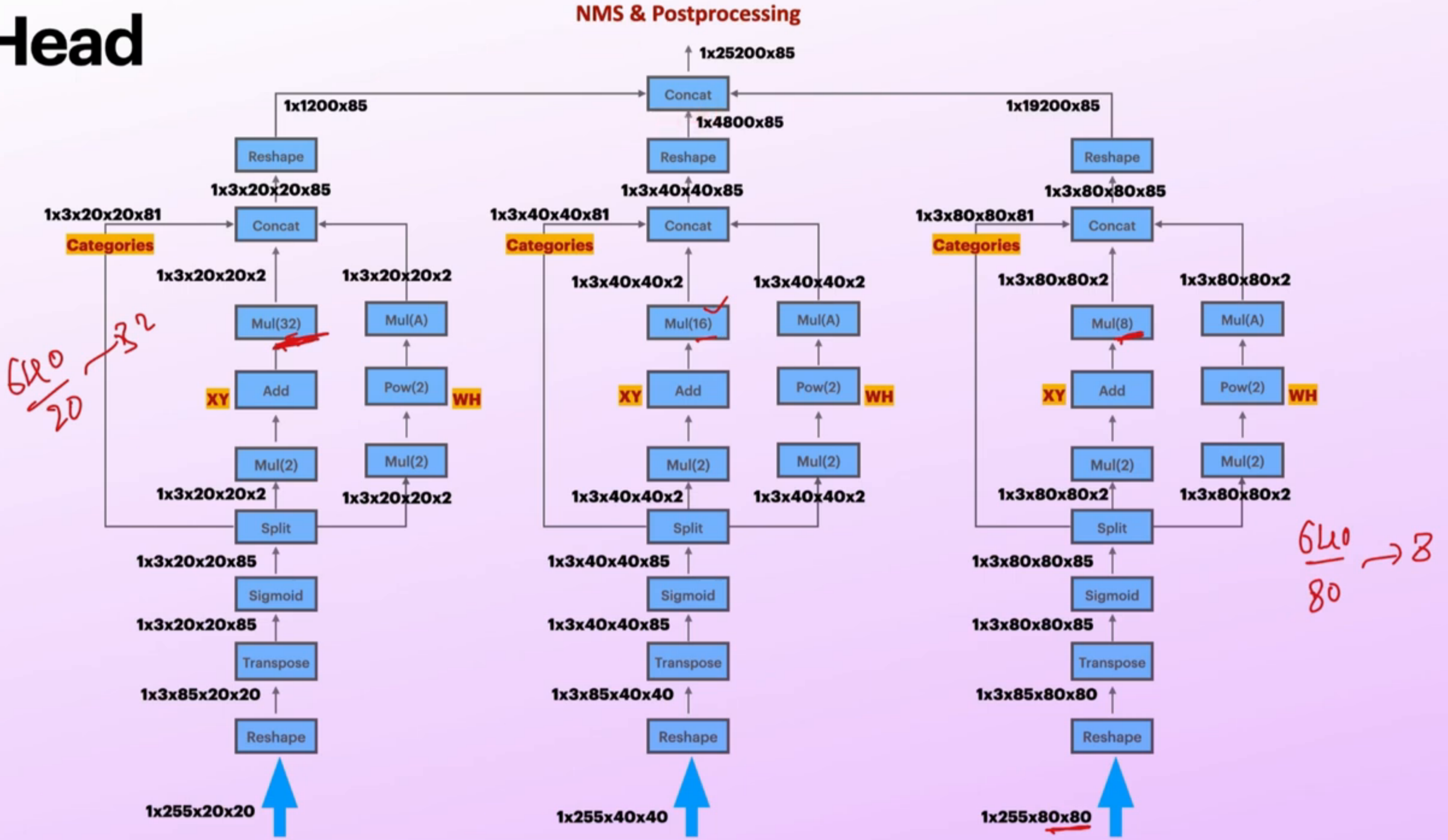
Head


NMS & Postprocessing
1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Reshape
Reshape
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
1x3x20x20x81
1x3x40x40x81
1x3x80x80x81
Concat
Concat
Concat
Categories
Categories
Categories
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Mul(32)
Mul(A)
Mul(16)
Mul(A)
Mul(8)
Mul(A)
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Split
Split
Split
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Sigmoid
Sigmoid
Sigmoid
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Transpose
Transpose
Transpose
1x3x85x20x20
1x3x85x40x40
1x3x85x80x80
Reshape
Reshape
Reshape
1x255x20x20
1x255x40x40
1x255x80x80
640/20 → 32
640/80 → 8

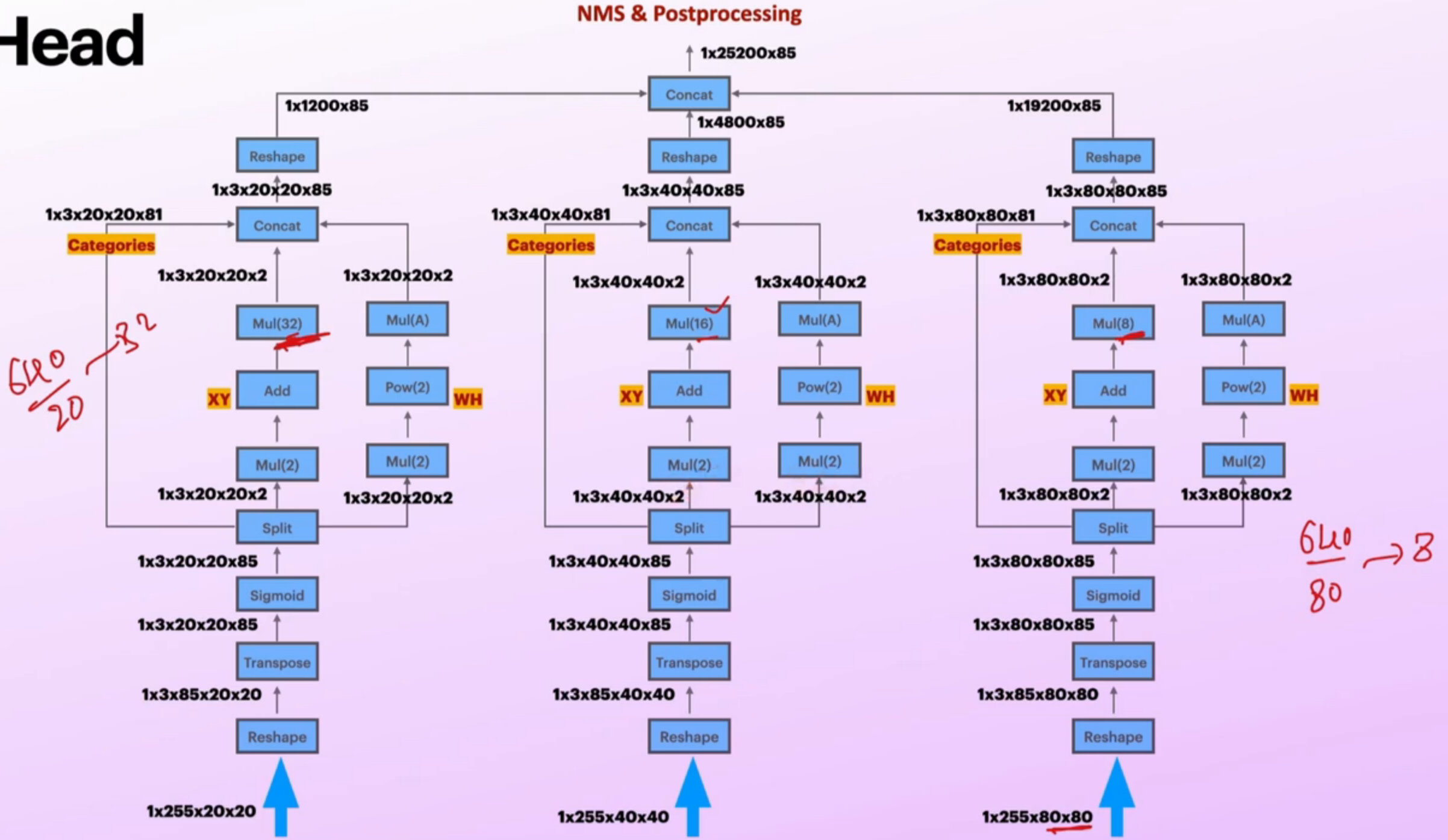
Head

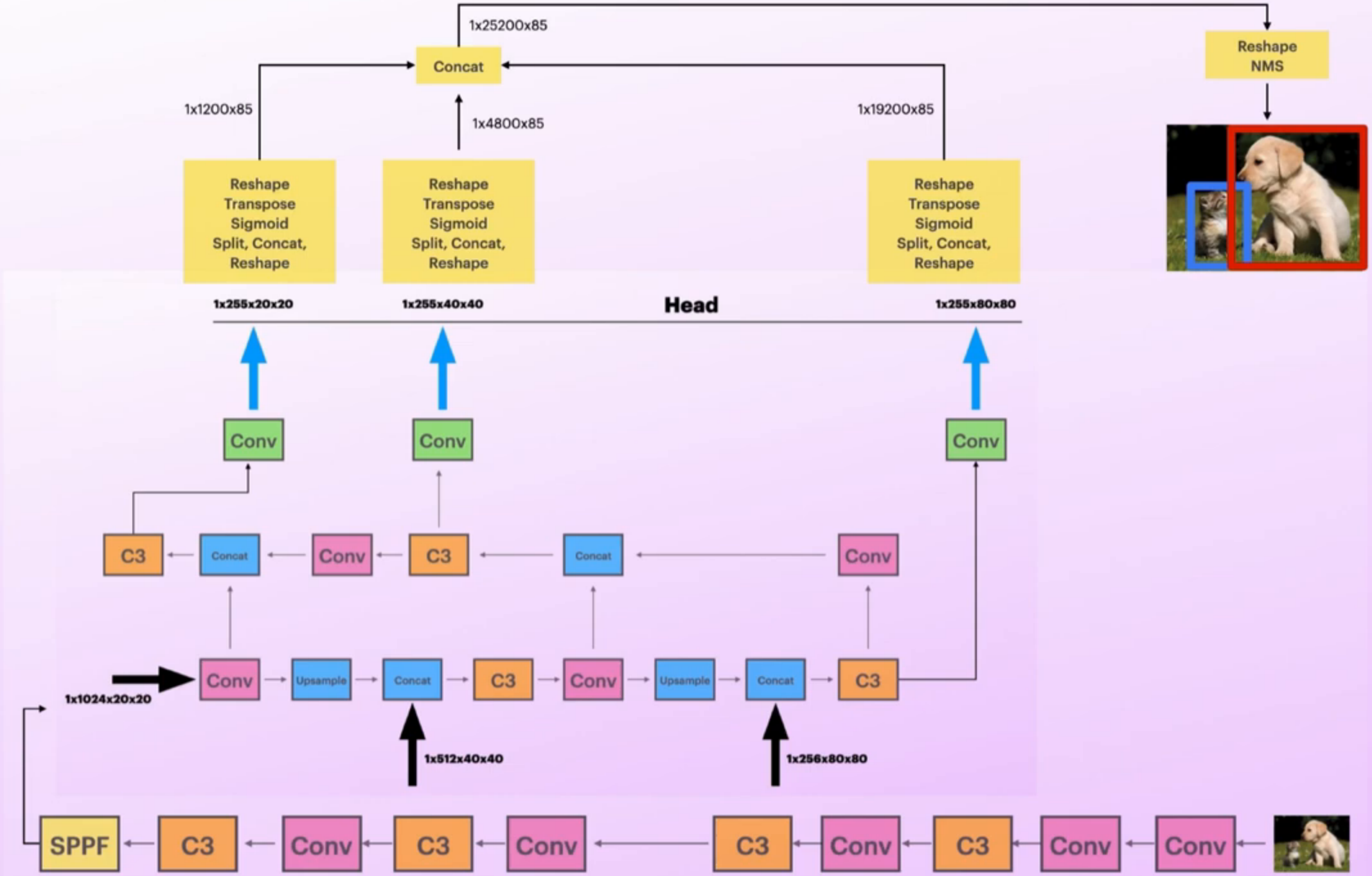

NMS & Postprocessing
1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Reshape
Reshape
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
1x3x20x20x81
1x3x40x40x81
1x3x80x80x81
Concat
Concat
Concat
Categories
Categories
Categories
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Mul(32)
Mul(A)
Mul(16)
Mul(A)
Mul(8)
Mul(A)
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Split
Split
Split
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Sigmoid
Sigmoid
Sigmoid
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Transpose
Transpose
Transpose
1x3x85x20x20
1x3x85x40x40
1x3x85x80x80
Reshape
Reshape
Reshape
1x255x20x20
1x255x40x40
1x255x80x80
640/20 → 32
640/80 → 8

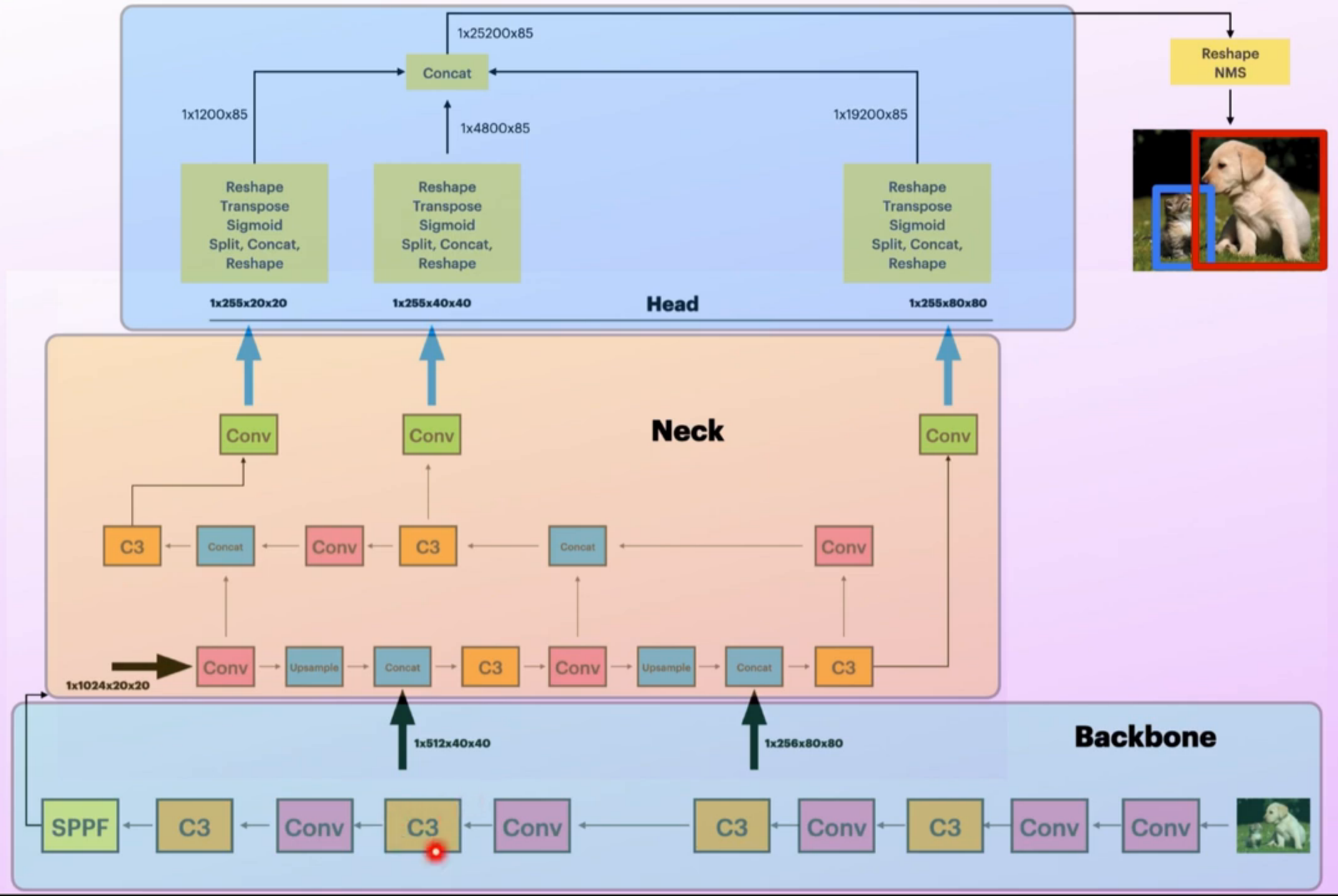
Head

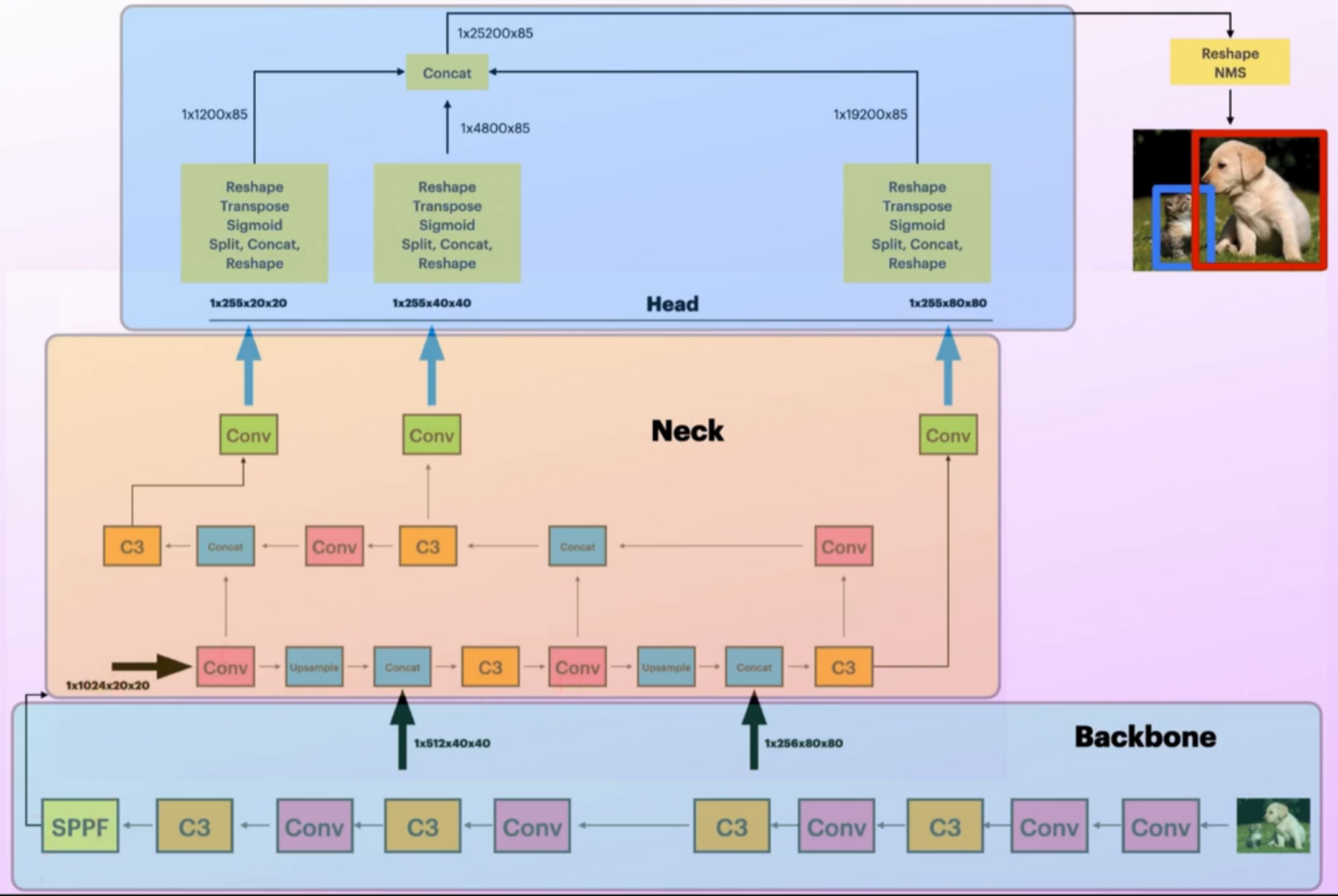

NMS & Postprocessing
1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Reshape
Reshape
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
1x3x20x20x81
1x3x40x40x81
1x3x80x80x81
Concat
Concat
Concat
Categories
Categories
Categories
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Mul(32)
Mul(A)
Mul(16)
Mul(A)
Mul(8)
Mul(A)
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Split
Split
Split
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Sigmoid
Sigmoid
Sigmoid
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Transpose
Transpose
Transpose
1x3x85x20x20
1x3x85x40x40
1x3x85x80x80
Reshape
Reshape
Reshape
1x255x20x20
1x255x40x40
1x255x80x80
640/20 → 32
640/80 → 8

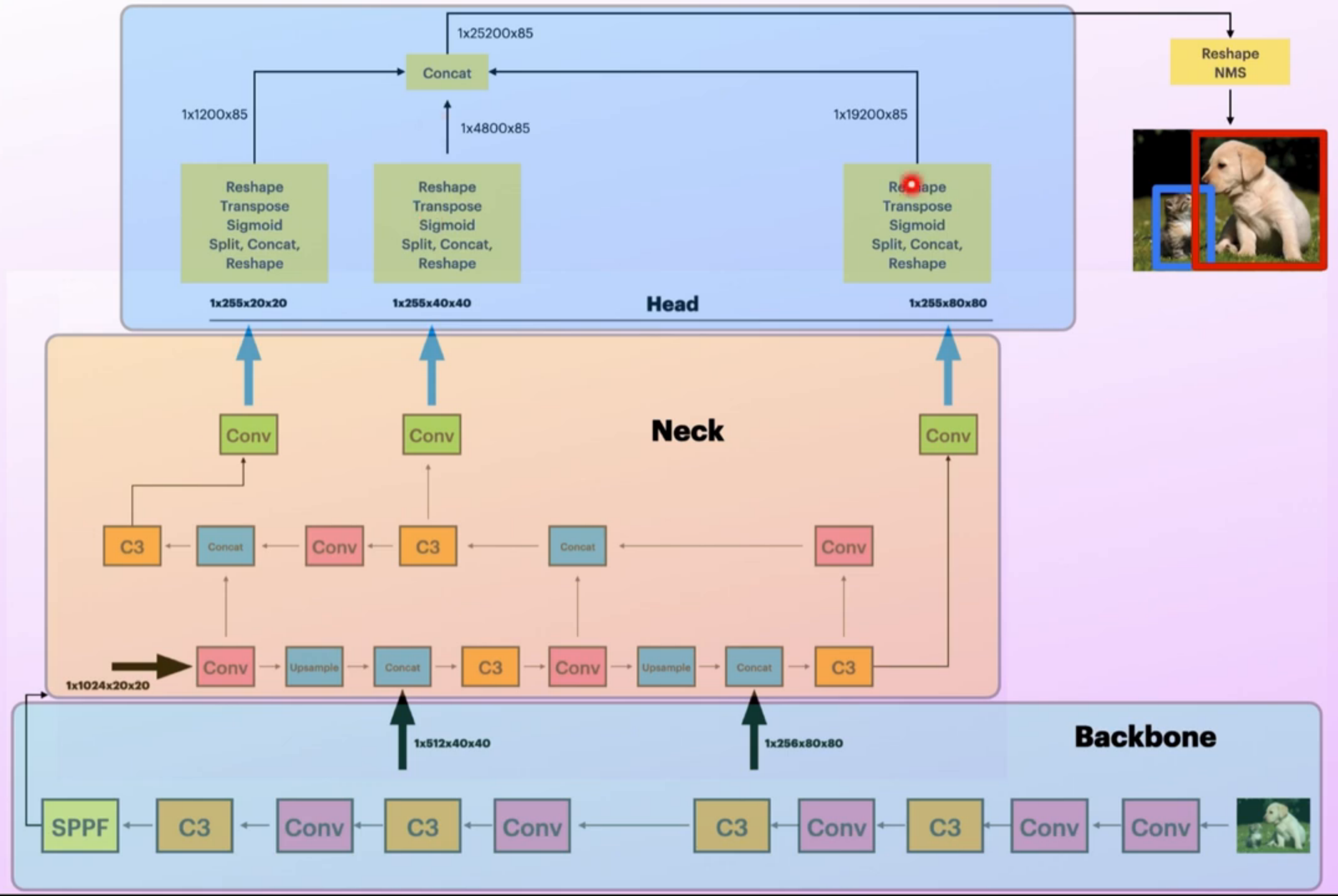
Head

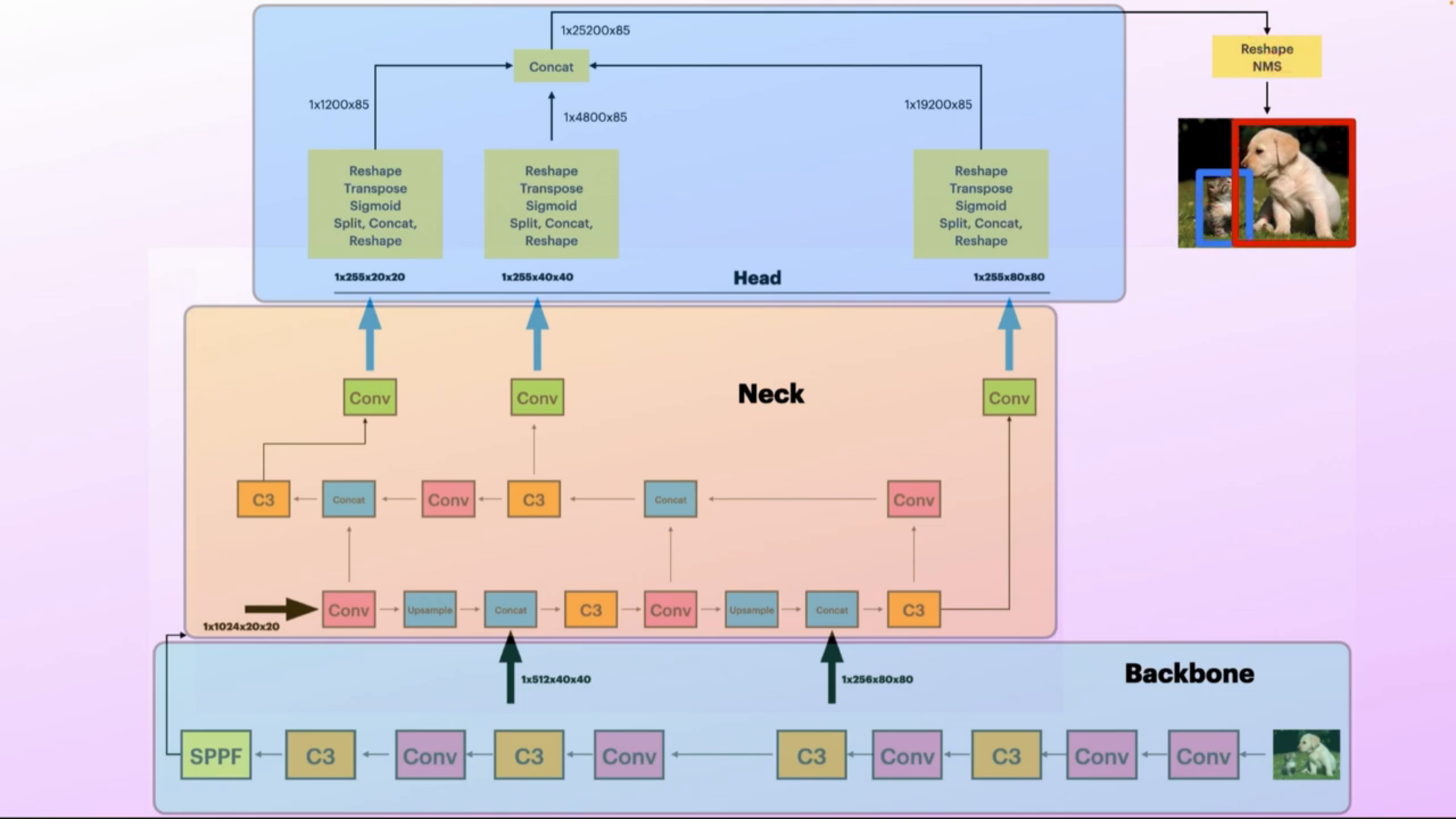

NMS & Postprocessing
1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Reshape
Reshape
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
1x3x20x20x81
1x3x40x40x81
1x3x80x80x81
Concat
Concat
Concat
Categories
Categories
Categories
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Mul(32)
Mul(A)
Mul(16)
Mul(A)
Mul(8)
Mul(A)
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
XY
Add
Pow(2)
WH
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
Mul(2)
1x3x20x20x2
1x3x20x20x2
1x3x40x40x2
1x3x40x40x2
1x3x80x80x2
1x3x80x80x2
Split
Split
Split
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Sigmoid
Sigmoid
Sigmoid
1x3x20x20x85
1x3x40x40x85
1x3x80x80x85
Transpose
Transpose
Transpose
1x3x85x20x20
1x3x85x40x40
1x3x85x80x80
Reshape
Reshape
Reshape
1x255x20x20
1x255x40x40
1x255x80x80
640/20 → 32
640/80 → 8

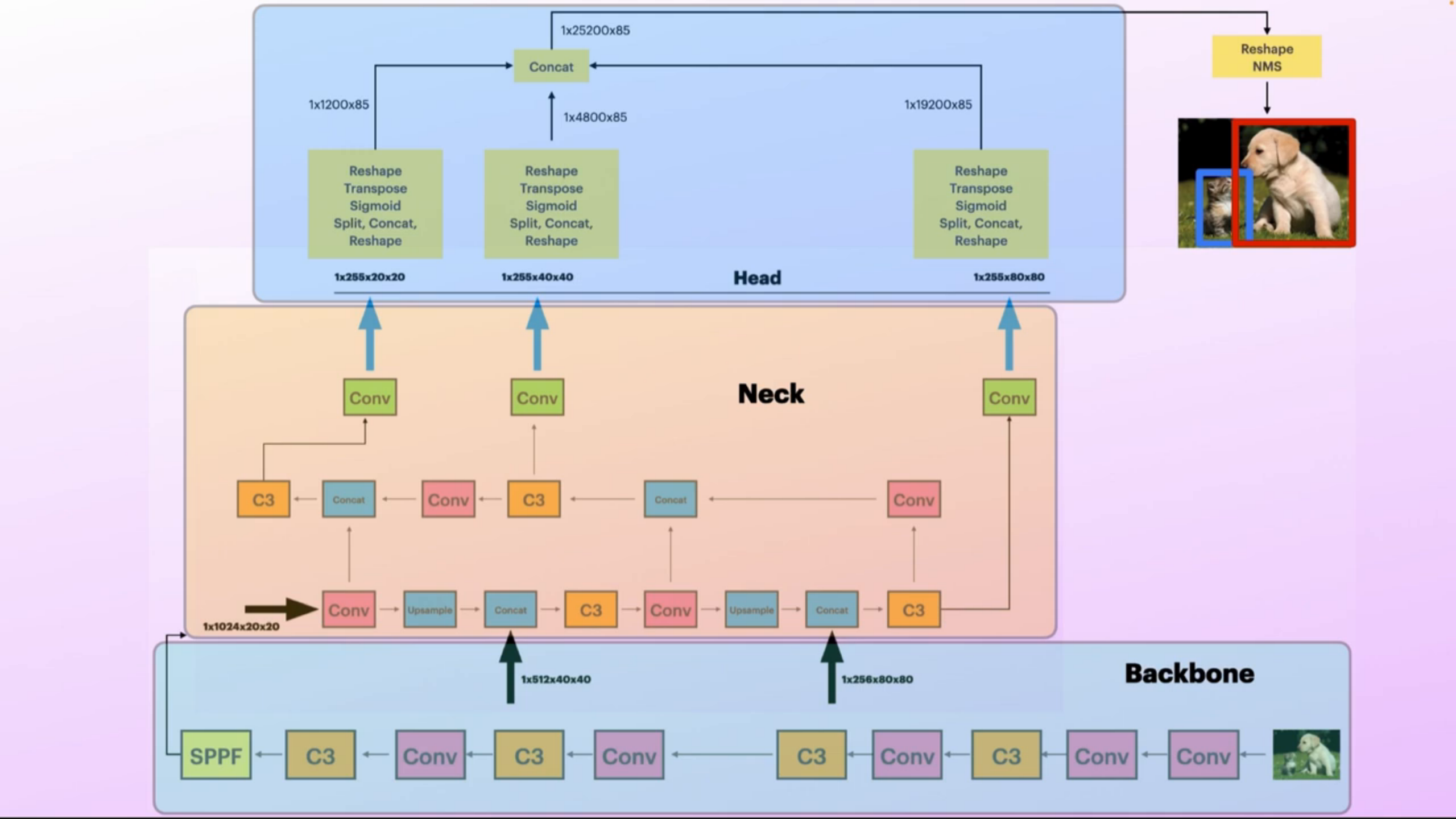

1x25200x85
Concat
Reshape
NMS
1x1200x85
1x4800x85
1x19200x85
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
1x255x20x20
1x255x40x40
Head
1x255x80x80
Conv
Conv
Conv
C3
Concat
Conv
C3
Concat
Conv
Conv
Upsample
Concat
C3
Conv
Upsample
Concat
C3
1x1024x20x20
1x512x40x40
1x256x80x80
SPPF
C3
Conv
C3
Conv
C3
Conv
C3
Conv
Conv

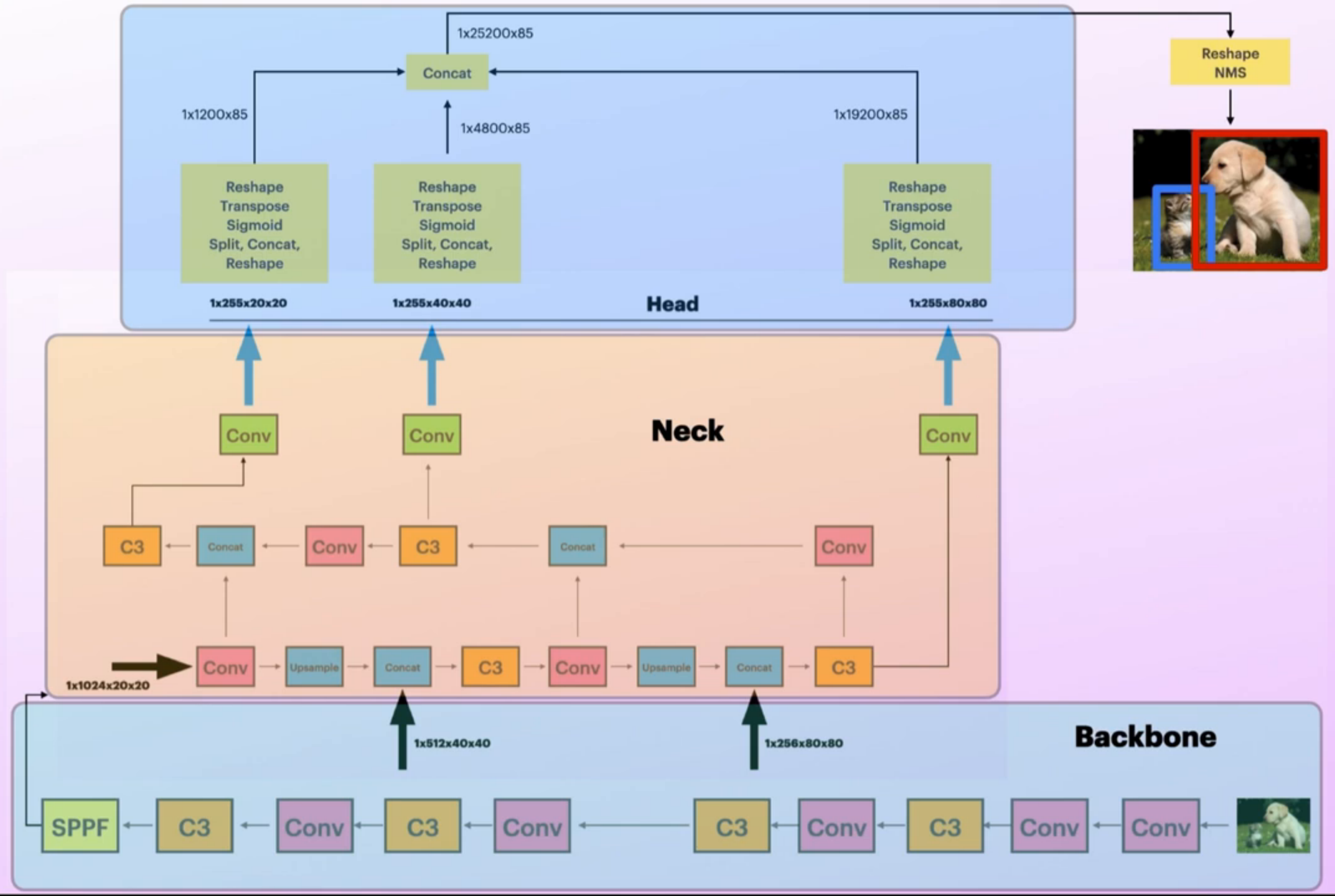

1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
1x255x20x20
1x255x40x40
Head
1x255x80x80
Reshape
NMS
Conv
Conv
Neck
Conv
C3
Concat
Conv
C3
Concat
Conv
Conv
Upsample
Concat
C3
Conv
Upsample
Concat
C3
1x1024x20x20
1x512x40x40
1x256x80x80
Backbone
SPPF
C3
Conv
C3
Conv
C3
Conv
C3
Conv
Conv


1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
1x255x20x20
1x255x40x40
Head
1x255x80x80
Reshape
NMS
Conv
Conv
Neck
Conv
C3
Concat
Conv
C3
Concat
Conv
Conv
Upsample
Concat
C3
Conv
Upsample
Concat
C3
1x1024x20x20
1x512x40x40
1x256x80x80
Backbone
SPPF
C3
Conv
C3
Conv
C3
Conv
C3
Conv
Conv


1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
1x255x20x20
1x255x40x40
Head
1x255x80x80
Reshape
NMS
Neck
Conv
Conv
Conv
C3
Concat
Conv
C3
Concat
Conv
Conv
Upsample
Concat
C3
Conv
Upsample
Concat
C3
1x1024x20x20
1x512x40x40
1x256x80x80
Backbone
SPPF
C3
Conv
C3
Conv
C3
Conv
C3
Conv
Conv


1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
1x255x20x20
1x255x40x40
Head
1x255x80x80
Reshape
NMS
Conv
Conv
Neck
Conv
C3
Concat
Conv
C3
Concat
Conv
Conv
Upsample
Concat
C3
Conv
Upsample
Concat
C3
1x1024x20x20
1x512x40x40
1x256x80x80
Backbone
SPPF
C3
Conv
C3
Conv
C3
Conv
C3
Conv
Conv


1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
1x255x20x20
1x255x40x40
Head
1x255x80x80
Reshape
NMS
Conv
Conv
Neck
Conv
C3
Concat
Conv
C3
Concat
Conv
Conv
Upsample
Concat
C3
Conv
Upsample
Concat
C3
1x1024x20x20
1x512x40x40
1x256x80x80
Backbone
SPPF
C3
Conv
C3
Conv
C3
Conv
C3
Conv
Conv


1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
1x255x20x20
1x255x40x40
Head
1x255x80x80
Reshape
NMS
Conv
Conv
Neck
Conv
C3
Concat
Conv
C3
Concat
Conv
Conv
Upsample
Concat
C3
Conv
Upsample
Concat
C3
1x1024x20x20
1x512x40x40
1x256x80x80
Backbone
SPPF
C3
Conv
C3
Conv
C3
Conv
C3
Conv
Conv

```
2  print(x2.shape)
3  model.model[1]
```

[101] Python

```
torch.Size([1, 64, 160, 160])
```

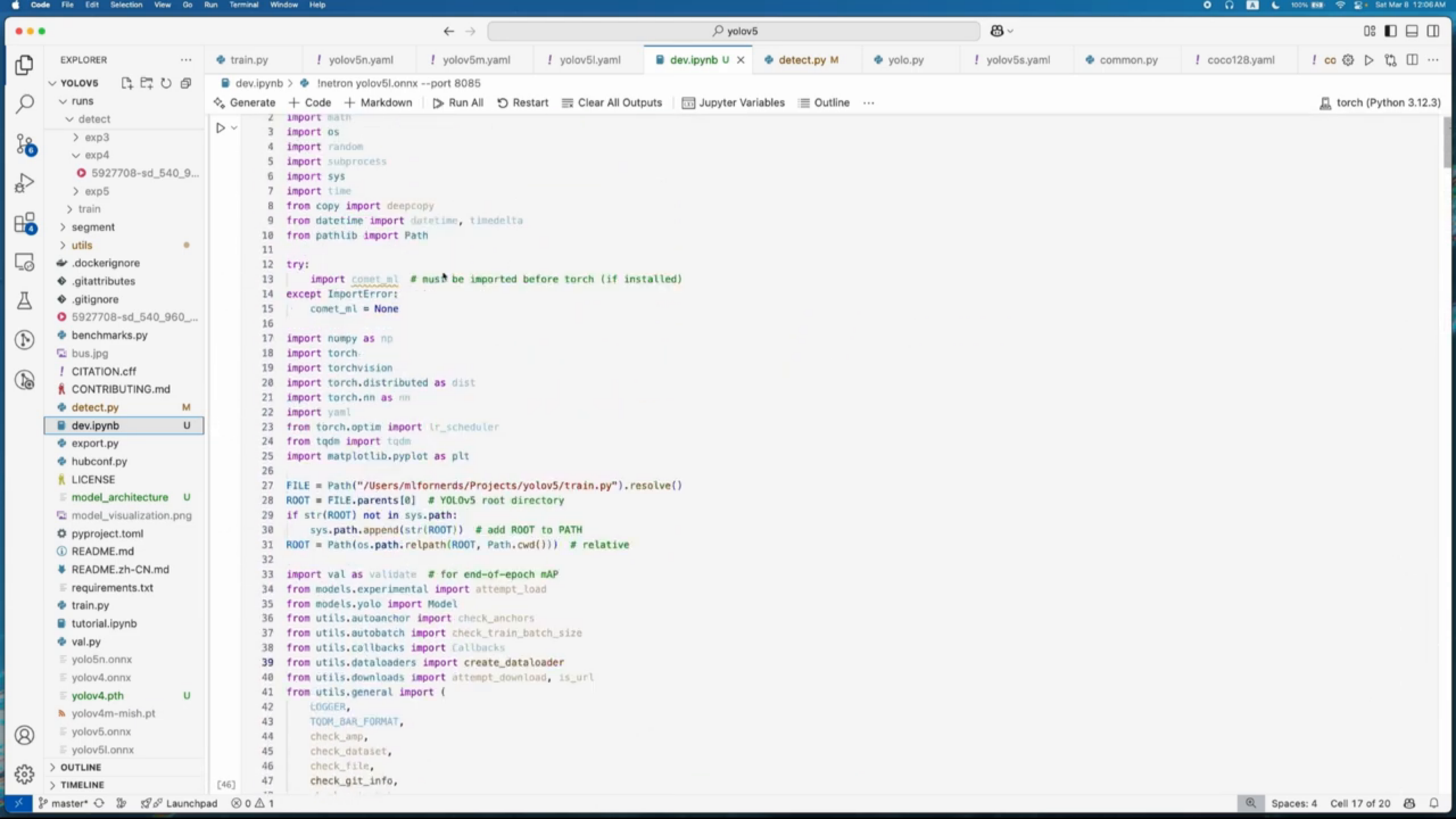
```
Conv(
  (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
  (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

```
1  x3 = model.model[2](x2)
2  print(x3.shape)
3  model.model[2]
```

[102] Python

```
torch.Size([1, 64, 160, 160])
```

```
C3(
  (cv1): Conv(
    (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (cv2): Conv(
    (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (cv3): Conv(
    (conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (m): Sequential(
    (0): Bottleneck(
      (cv1): Conv(
        (conv): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
...
```


yolov5
train.py
yolov5n.yaml
yolov5m.yaml
yolov5l.yaml
dev.ipynb U
detect.py M
yolo.py
yolov5s.yaml
common.py
coco128.yaml
EXPLORER
YOLOV5
runs
detect
exp3
exp4
5927708-sd_540_9...
exp5
train
segment
utils
.dockerignore
.gitattributes
.gitignore
5927708-sd_540_960_...
benchmarks.py
bus.jpg
CITATION.cff
CONTRIBUTING.md
detect.py M
dev.ipynb U
export.py
hubconf.py
LICENSE
model_architecture U
model_visualization.png
pyproject.toml
README.md
README.zh-CN.md
requirements.txt
train.py
tutorial.ipynb
val.py
yolo5n.onnx
yolov4.onnx
yolov4.pth U
yolov4m-mish.pt
yolov5.onnx
yolov5l.onnx
OUTLINE
TIMELINE
dev.ipynb > !netron yolov5l.onnx --port 8085
Generate
Code
Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline
torch (Python 3.12.3)
import math
import os
import random
import subprocess
import sys
import time
from copy import deepcopy
from datetime import datetime, timedelta
from pathlib import Path
try:
import comet_ml # must be imported before torch (if installed)
except ImportError:
comet_ml = None
import numpy as np
import torch
import torchvision
import torch.distributed as dist
import torch.nn as nn
import yaml
from torch.optim import lr_scheduler
from tqdm import tqdm
import matplotlib.pyplot as plt
FILE = Path("/Users/mlfornerds/Projects/yolov5/train.py").resolve()
ROOT = FILE.parents[0] # YOLOv5 root directory
if str(ROOT) not in sys.path:
sys.path.append(str(ROOT)) # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd())) # relative
import val as validate # for end-of-epoch mAP
from models.experimental import attempt_load
from models.yolo import Model
from utils.autoanchor import check_anchors
from utils.autobatch import check_train_batch_size
from utils.callbacks import Callbacks
from utils.dataloaders import create_dataloader
from utils.downloads import attempt_download, is_url
from utils.general import (
LOGGER,
TQDM_BAR_FORMAT,
check_amp,
check_dataset,
check_file,
check_git_info,
[46]
master*
Launchpad
Spaces: 4
Cell 17 of 20

```
X = X.to(device).float() / 255 # image, converted to proper range
print(X.shape)
preds = model(X)
print(len(preds))
preds[0].shape, preds[1].shape, preds[2].shape
```

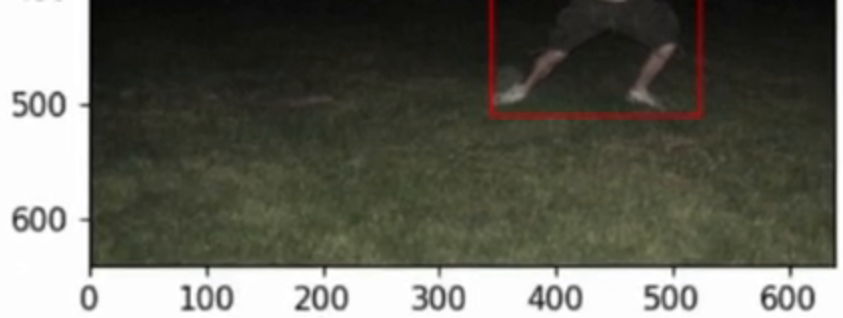
[99] Python

```
                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12          [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16          [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1    229245  models.yolo.Detect                      [80, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]], [128, 256, 512]]
YOLOv5s summary: 214 layers, 7235389 parameters, 7235389 gradients, 16.6 GFLOPs

torch.Size([1, 3, 640, 640])
3

(torch.Size([1, 3, 80, 80, 85]),
 torch.Size([1, 3, 40, 40, 85]),
 torch.Size([1, 3, 20, 20, 85]))
```

```
x1 = model.model[0](X)
```




```
[ ]
1
```

(parameter) anchors: Any | None

```
[99]
1 model = Model(cfg, ch=3, nc=80, anchors=hyp.get('anchors')).to(device)
2 X = X.to(device).float() / 255 # image, converted to proper range
3 print(X.shape)
4 preds = model(X)
5 print(len(preds))
6 preds[0].shape, preds[1].shape, preds[2].shape
```

```
...
                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
```

```
329                 math.log(0.6 / (m.nc - 0.99999)) if cf is None else torch.log(cf / cf.sum())
330             )  # cls
331             mi.bias = torch.nn.Parameter(b.view(-1), requires_grad=True)
332
333
334 Model = DetectionModel  # retain YOLOv5 'Model' class for backwards compatibility      Glenn Jocher, 3 years ago • New YOLOv5 Classification Models
335
336
    Glenn Jocher, 6 months ago | 3 authors (Glenn Jocher and others)
337 class SegmentationModel(DetectionModel):
338     """YOLOv5 segmentation model for object detection and segmentation tasks with configurable parameters."""
339
340     def __init__(self, cfg="yolov5s-seg.yaml", ch=3, nc=None, anchors=None):
341         """Initializes a YOLOv5 segmentation model with configurable params: cfg (str) for configuration, ch (int) for channels, nc (int) for num c
342         super().__init__(cfg, ch, nc, anchors)
343
344
    Glenn Jocher, 6 months ago | 2 authors (Glenn Jocher and one other)
345 class ClassificationModel(BaseModel):
346     """YOLOv5 classification model for image classification tasks, initialized with a config file or detection model."""
347
348     def __init__(self, cfg=None, model=None, nc=1000, cutoff=10):
349         """Initializes YOLOv5 model with config file `cfg`, input channels `ch`, number of classes `nc`, and `cuttoff`
350         index.
351         """
352         super().__init__()
353         self._from_detection_model(model, nc, cutoff) if model is not None else self._from_yaml(cfg)
354
355     def _from_detection_model(self, model, nc=1000, cutoff=10):
356         """Creates a classification model from a YOLOv5 detection model, slicing at `cutoff` and adding a classification
357         layer.
358         """
359         if isinstance(model, DetectMultiBackend):
360             model = model.model  # unwrap DetectMultiBackend
361         model.model = model.model[:cutoff]  # backbone
362         m = model.model[-1]  # last layer
363         ch = m.conv.in_channels if hasattr(m, "conv") else m.cv1.conv.in_channels  # ch into module
364         c = Classify(ch, nc)  # Classify()
365         c.i, c.f, c.type = m.i, m.f, "models.common.Classify"  # index, from, type
```

dels
_pycache__
ub
egment
_init__.py
ommon.py
xperimental.py
.py
lo.py
lov5l.yaml
lov5m.yaml
lov5n.yaml
lov5s.yaml
lov5x.yaml
s
etect
exp
exp2
exp3
exp4
5927708-sd_540_9...
exp5
ain
gment
s
ckerignore
attributes
ignore
27708-sd_540_960_...
INE
LINE

```
217
        Glenn Jocher, 6 months ago | 5 authors (Glenn Jocher and others)
218     class DetectionModel(BaseModel):
219         """YOLOv5 detection model class for object detection tasks, supporting custom configurations and anchors."""
220
221         def __init__(self, cfg="yolov5s.yaml", ch=3, nc=None, anchors=None):      Glenn Jocher, 12 months ago • Replace inline comments with docstrings
222             """Initializes YOLOv5 model with configuration file, input channels, number of classes, and custom anchors."""
223             super().__init__()
224             if isinstance(cfg, dict):
225                 self.yaml = cfg  # model dict
226             else:  # is *.yaml
227                 import yaml  # for torch hub
228
229                 self.yaml_file = Path(cfg).name
230                 with open(cfg, encoding="ascii", errors="ignore") as f:
231                     self.yaml = yaml.safe_load(f)  # model dict
232
233             # Define model
234             ch = self.yaml["ch"] = self.yaml.get("ch", ch)  # input channels
235             if nc and nc != self.yaml["nc"]:
236                 LOGGER.info(f"Overriding model.yaml nc={self.yaml['nc']} with nc={nc}")
237                 self.yaml["nc"] = nc  # override yaml value
238             if anchors:
239                 LOGGER.info(f"Overriding model.yaml anchors with anchors={anchors}")
240                 self.yaml["anchors"] = round(anchors)  # override yaml value
241             self.model, self.save = parse_model(deepcopy(self.yaml), ch=[ch])  # model, savelist
242             self.names = [str(i) for i in range(self.yaml["nc"])]  # default names
243             self.inplace = self.yaml.get("inplace", True)
244
245             # Build strides, anchors
246             m = self.model[-1]  # Detect()
247             if isinstance(m, (Detect, Segment)):
248
249                 def _forward(x):
250                     """Passes the input 'x' through the model and returns the processed output."""
251                     return self.forward(x)[0] if isinstance(m, Segment) else self.forward(x)
252
253                 s = 256  # 2x min stride
```

```python
model = Model(cfg, ch=3, nc=80, anchors=hyp.get('anchors')).to(device)
X = X.to(device).float() / 255 # image, converted to proper range
print(X.shape)
preds = model(X)
print(len(preds))
preds[0].shape, preds[1].shape, preds[2].shape
```

[99]

...

	from	n	params	module	arguments
0	-1	1	3520	models.common.Conv	[3, 32, 6, 2, 2]
1	-1	1	18560	models.common.Conv	[32, 64, 3, 2]
2	-1	1	18816	models.common.C3	[64, 64, 1]
3	-1	1	73984	models.common.Conv	[64, 128, 3, 2]
4	-1	2	115712	models.common.C3	[128, 128, 2]
5	-1	1	295424	models.common.Conv	[128, 256, 3, 2]
6	-1	3	625152	models.common.C3	[256, 256, 3]
7	-1	1	1180672	models.common.Conv	[256, 512, 3, 2]
8	-1	1	1182720	models.common.C3	[512, 512, 1]
9	-1	1	656896	models.common.SPPF	[512, 512, 5]
10	-1	1	131584	models.common.Conv	[512, 256, 1, 1]
11	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
12	[-1, 6]	1	0	models.common.Concat	[1]
13	-1	1	361984	models.common.C3	[512, 256, 1, False]
14	-1	1	33024	models.common.Conv	[256, 128, 1, 1]
15	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
16	[-1, 4]	1	0	models.common.Concat	[1]
17	-1	1	90880	models.common.C3	[256, 128, 1, False]
18	-1	1	147712	models.common.Conv	[128, 128, 3, 2]
19	[-1, 14]	1	0	models.common.Concat	[1]
20	-1	1	296448	models.common.C3	[256, 256, 1, False]
21	-1	1	590336	models.common.Conv	[256, 256, 3, 2]
22	[-1, 10]	1	0	models.common.Concat	[1]
23	-1	1	1182720	models.common.C3	[512, 512, 1, False]
24	[17, 20, 23]	1	229245	models.yolo.Detect	[80, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156,

```
YOLOv5s summary: 214 layers, 7235389 parameters, 7235389 gradients, 16.6 GFLOPs

torch.Size([1, 3, 640, 640])
```

```
model = Model(cfg, ch=3, nc=80, anchors=hyp.get('anchors')).to(device)
X = X.to(device).float() / 255 # image, converted to proper range
print(X.shape)
preds = model(X)
print(len(preds))
preds[0].shape, preds[1].shape, preds[2].shape
```

[99]

...

	from	n	params	module	arguments
0	-1	1	3520	models.common.Conv	[3, 32, 6, 2, 2]
1	-1	1	18560	models.common.Conv	[32, 64, 3, 2]
2	-1	1	18816	models.common.C3	[64, 64, 1]
3	-1	1	73984	models.common.Conv	[64, 128, 3, 2]
4	-1	2	115712	models.common.C3	[128, 128, 2]
5	-1	1	295424	models.common.Conv	[128, 256, 3, 2]
6	-1	3	625152	models.common.C3	[256, 256, 3]
7	-1	1	1180672	models.common.Conv	[256, 512, 3, 2]
8	-1	1	1182720	models.common.C3	[512, 512, 1]
9	-1	1	656896	models.common.SPPF	[512, 512, 5]
10	-1	1	131584	models.common.Conv	[512, 256, 1, 1]
11	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
12	[-1, 6]	1	0	models.common.Concat	[1]
13	-1	1	361984	models.common.C3	[512, 256, 1, False]
14	-1	1	33024	models.common.Conv	[256, 128, 1, 1]
15	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
16	[-1, 4]	1	0	models.common.Concat	[1]
17	-1	1	90880	models.common.C3	[256, 128, 1, False]
18	-1	1	147712	models.common.Conv	[128, 128, 3, 2]
19	[-1, 14]	1	0	models.common.Concat	[1]
20	-1	1	296448	models.common.C3	[256, 256, 1, False]
21	-1	1	590336	models.common.Conv	[256, 256, 3, 2]
22	[-1, 10]	1	0	models.common.Concat	[1]
23	-1	1	1182720	models.common.C3	[512, 512, 1, False]
24	[17, 20, 23]	1	229245	models.yolo.Detect	[80, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156,

```
YOLOv5s summary: 214 layers, 7235389 parameters, 7235389 gradients, 16.6 GFLOPs

torch.Size([1, 3, 640, 640])
```

```python
model = Model(cfg, ch=3, nc=80, anchors=hyp.get('anchors')).to(device)
X = X.to(device).float() / 255 # image, converted to proper range
print(X.shape)
preds = model(X)
print(len(preds))
preds[0].shape, preds[1].shape, preds[2].shape
```

[99]

...

	from	n	params	module	arguments
0	-1	1	3520	models.common.Conv	[3, 32, 6, 2, 2]
1	-1	1	18560	models.common.Conv	[32, 64, 3, 2]
2	-1	1	18816	models.common.C3	[64, 64, 1]
3	-1	1	73984	models.common.Conv	[64, 128, 3, 2]
4	-1	2	115712	models.common.C3	[128, 128, 2]
5	-1	1	295424	models.common.Conv	[128, 256, 3, 2]
6	-1	3	625152	models.common.C3	[256, 256, 3]
7	-1	1	1180672	models.common.Conv	[256, 512, 3, 2]
8	-1	1	1182720	models.common.C3	[512, 512, 1]
9	-1	1	656896	models.common.SPPF	[512, 512, 5]
10	-1	1	131584	models.common.Conv	[512, 256, 1, 1]
11	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
12	[-1, 6]	1	0	models.common.Concat	[1]
13	-1	1	361984	models.common.C3	[512, 256, 1, False]
14	-1	1	33024	models.common.Conv	[256, 128, 1, 1]
15	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
16	[-1, 4]	1	0	models.common.Concat	[1]
17	-1	1	90880	models.common.C3	[256, 128, 1, False]
18	-1	1	147712	models.common.Conv	[128, 128, 3, 2]
19	[-1, 14]	1	0	models.common.Concat	[1]
20	-1	1	296448	models.common.C3	[256, 256, 1, False]
21	-1	1	590336	models.common.Conv	[256, 256, 3, 2]
22	[-1, 10]	1	0	models.common.Concat	[1]
23	-1	1	1182720	models.common.C3	[512, 512, 1, False]
24	[17, 20, 23]	1	229245	models.yolo.Detect	[80, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156,

```
YOLOv5s summary: 214 layers, 7235389 parameters, 7235389 gradients, 16.6 GFLOPs

torch.Size([1, 3, 640, 640])
```

```
YOLOv5s summary: 214 layers, 7235389 parameters, 7235389 gradients, 16.6 GFLOPs

torch.Size([1, 3, 640, 640])
3

(torch.Size([1, 3, 80, 80, 85]),
 torch.Size([1, 3, 40, 40, 85]),
 torch.Size([1, 3, 20, 20, 85]))
```

[100]
```
x1 = model.model[0](X)
print(x1.shape)
model.model[0]
```

```
torch.Size([1, 32, 320, 320])

Conv(
  (conv): Conv2d(3, 32, kernel_size=(6, 6), stride=(2, 2), padding=(2, 2), bias=False)
  (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

[101]
```
x2 = model.model[1](x1)
print(x2.shape)
model.model[1]
```

```
torch.Size([1, 64, 160, 160])

Conv(
  (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
  (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

ckerignore
attributes
ignore
27708-sd_540_960_...
nchmarks.py
s.jpg
TATION.cff
NTRIBUTING.md
ect.py M
v.ipynb U
port.py
oconf.py
ENSE
del_architecture U
del_visualization.png
project.toml
ADME.md
ADME.zh-CN.md
uirements.txt
n.py
orial.ipynb
.py
o5n.onnx
ov4.onnx
ov4.pth U
ov4m-mish.pt
ov5.onnx
ov5l.onnx

```
YOLOv5s summary: 214 layers, 7235389 parameters, 7235389 gradients, 16.6 GFLOPs

torch.Size([1, 3, 640, 640])
3

(torch.Size([1, 3, 80, 80, 85]),
 torch.Size([1, 3, 40, 40, 85]),
 torch.Size([1, 3, 20, 20, 85]))
```

[100]
```
x1 = model.model[0](X)
print(x1.shape)
model.model[0]
```

```
torch.Size([1, 32, 320, 320])

Conv(
  (conv): Conv2d(3, 32, kernel_size=(6, 6), stride=(2, 2), padding=(2, 2), bias=False)
  (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

[101]
```
x2 = model.model[1](x1)
print(x2.shape)
model.model[1]
```

```
torch.Size([1, 64, 160, 160])

Conv(
  (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
  (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

```
YOLOv5s summary: 214 layers, 7235389 parameters, 7235389 gradients, 16.6 GFLOPs

torch.Size([1, 3, 640, 640])
3

(torch.Size([1, 3, 80, 80, 85]),
 torch.Size([1, 3, 40, 40, 85]),
 torch.Size([1, 3, 20, 20, 85]))
```

[100]

```
x1 = model.model[0](X)
print(x1.shape)
model.model[0]
```

```
torch.Size([1, 32, 320, 320])

Conv(
  (conv): Conv2d(3, 32, kernel_size=(6, 6), stride=(2, 2), padding=(2, 2), bias=False)
  (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

[101]

```
x2 = model.model[1](x1)
print(x2.shape)
model.model[1]
```

```
torch.Size([1, 64, 160, 160])

Conv(
  (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
  (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

```
YOLOv5s summary: 214 layers, 7235389 parameters, 7235389 gradients, 16.6 GFLOPs

torch.Size([1, 3, 640, 640])
3

(torch.Size([1, 3, 80, 80, 85]),
 torch.Size([1, 3, 40, 40, 85]),
 torch.Size([1, 3, 20, 20, 85]))
```

```
[100]
1 x1 = model.model[0](X)
2 print(x1.shape)
3 model.model[0]
```

```
torch.Size([1, 32, 320, 320])

Conv(
  (conv): Conv2d(3, 32, kernel_size=(6, 6), stride=(2, 2), padding=(2, 2), bias=False)
  (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

```
[101]
1 x2 = model.model[1](x1)
2 print(x2.shape)
3 model.model[1]
```

```
torch.Size([1, 64, 160, 160])

Conv(
  (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
  (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

Launchpad ⊗ 0 ⚠ 1

s
ckerignore
attributes
ignore
27708-sd_540_960_...
nchmarks.py
s.jpg
TATION.cff
NTRIBUTING.md
ect.py M
y.ipynb U
port.py
oconf.py
ENSE
del_architecture U
del_visualization.png
project.toml
ADME.md
ADME.zh-CN.md
uirements.txt
n.py
orial.ipynb
.py
o5n.onnx
ov4.onnx
ov4.pth U
ov4m-mish.pt
ov5.onnx
ov5l.onnx
INE
LINE

```
YOLOv5s summary: 214 layers, 7235389 parameters, 7235389 gradients, 16.6 GFLOPs

torch.Size([1, 3, 640, 640])
3

(torch.Size([1, 3, 80, 80, 85]),
 torch.Size([1, 3, 40, 40, 85]),
 torch.Size([1, 3, 20, 20, 85]))
```

[100]

```
x1 = model.model[0](X)
print(x1.shape)
model.model[0]
```

```
torch.Size([1, 32, 320, 320])

Conv(
  (conv): Conv2d(3, 32, kernel_size=(6, 6), stride=(2, 2), padding=(2, 2), bias=False)
  (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

[101]

```
x2 = model.model[1](x1)
print(x2.shape)
model.model[1]
```

```
torch.Size([1, 64, 160, 160])

Conv(
  (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
  (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

```
YOLOv5s summary: 214 layers, 7235389 parameters, 7235389 gradients, 16.6 GFLOPs

torch.Size([1, 3, 640, 640])
3

(torch.Size([1, 3, 80, 80, 85]),
 torch.Size([1, 3, 40, 40, 85]),
 torch.Size([1, 3, 20, 20, 85]))
```

```
[100]
1  x1 = model.model[0](X)
2  print(x1.shape)
3  model.model[0]
```

```
torch.Size([1, 32, 320, 320])

Conv(
  (conv): Conv2d(3, 32, kernel_size=(6, 6), stride=(2, 2), padding=(2, 2), bias=False)
  (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

```
[101]
1  x2 = model.model[1](x1)
2  print(x2.shape)
3  model.model[1]
```

```
torch.Size([1, 64, 160, 160])

Conv(
  (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
  (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

```
torch.Size([1, 3, 20, 20, 85]))
```

```
x1 = model.model[0](X)
print(x1.shape)
model.model[0]
```

[100]

```
torch.Size([1, 32, 320, 320])

Conv(
  (conv): Conv2d(3, 32, kernel_size=(6, 6), stride=(2, 2), padding=(2, 2), bias=False)
  (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

```
x2 = model.model[1](x1)
print(x2.shape)
model.model[1]
```

[101]

```
torch.Size([1, 64, 160, 160])

Conv(
  (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
  (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
  (act): SiLU(inplace=True)
)
```

```
x3 = model.model[2](x2)
print(x3.shape)
model.model[2]
```

[102]

```
  (act): SiLU(inplace=True)
)
```

```
x3 = model.model[2](x2)
print(x3.shape)
model.model[2]
```

[102]

```
torch.Size([1, 64, 160, 160])

C3(
  (cv1): Conv(
    (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (cv2): Conv(
    (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (cv3): Conv(
    (conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (m): Sequential(
    (0): Bottleneck(
      (cv1): Conv(
        (conv): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
...
```

```
    (act): SiLU(inplace=True)
  )
  (cv2): Conv(
    (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (cv3): Conv(
    (conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (m): Sequential(
    (0): Bottleneck(
      (cv1): Conv(
        (conv): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
    )
  )
)
```

```
[104]
# Create dummy input with correct shape
dummy_input = torch.randn(1, 3, 640, 640)
dummy_input = dummy_input.to(device)
# Export to ONNX
torch.onnx.export(
    model,
    dummy_input,
    "yolov5s.onnx",
```

Launchpad ⊗ 0 ⚠ 1

```
torch.Size([1, 64, 160, 160])
```

```
C3(
  (cv1): Conv(
    (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (cv2): Conv(
    (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (cv3): Conv(
    (conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (m): Sequential(
    (0): Bottleneck(
      (cv1): Conv(
        (conv): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
    )
  )
```

```python
# Create dummy input with correct shape
dummy_input = torch.randn(1, 3, 640, 640)
dummy_input = dummy_input.to(device)
# Export to ONNX
```

```
torch.Size([1, 64, 160, 160])
```

```
C3(
  (cv1): Conv(
    (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (cv2): Conv(
    (conv): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (cv3): Conv(
    (conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (m): Sequential(
    (0): Bottleneck(
      (cv1): Conv(
        (conv): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
    )
  )
```

```python
# Create dummy input with correct shape
dummy_input = torch.randn(1, 3, 640, 640)
dummy_input = dummy_input.to(device)
# Export to ONNX
torch.onnx.export(
```

```
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (m): Sequential(
    (0): Bottleneck(
      (cv1): Conv(
        (conv): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
    )
  )
)
```

[104]

```python
# Create dummy input with correct shape
dummy_input = torch.randn(1, 3, 640, 640)
dummy_input = dummy_input.to(device)
# Export to ONNX
torch.onnx.export(
    model,
    dummy_input,
    "yolov5s.onnx",
    opset_version=12,
    # do_constant_folding=False,  # Prevent folding
    input_names=['input'],
    output_names=['output']
)
```

```
/Users/mlfornerds/Projects/yolov5/models/yolo.py:268: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if augment:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:171: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if visualize:
```

```
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (m): Sequential(
    (0): Bottleneck(
      (cv1): Conv(
        (conv): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
    )
  )
)
```

Generate | + Code | + Markdown

[104]

```python
# Create dummy input with correct shape
dummy_input = torch.randn(1, 3, 640, 640)
dummy_input = dummy_input.to(device)
# Export to ONNX
torch.onnx.export(
    model,
    dummy_input,
    "yolov5s.onnx",
    opset_version=12,
    # do_constant_folding=False,  # Prevent folding
    input_names=['input'],
    output_names=['output']
)
```

```
/Users/mlfornerds/Projects/yolov5/models/yolo.py:268: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if augment:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:171: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if visualize:
```

```
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (m): Sequential(
    (0): Bottleneck(
      (cv1): Conv(
        (conv): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
    )
  )
)
```

[104]

```python
# Create dummy input with correct shape
dummy_input = torch.randn(1, 3, 640, 640)
dummy_input = dummy_input.to(device)
# Export to ONNX
torch.onnx.export(
    model,
    dummy_input,
    "yolov5s.onnx",
    opset_version=12,
    # do_constant_folding=False,  # Prevent folding
    input_names=['input'],
    output_names=['output']
)
```

```
/Users/mlfornerds/Projects/yolov5/models/yolo.py:268: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if augment:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:171: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if visualize:
```

```
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
    (act): SiLU(inplace=True)
  )
  (m): Sequential(
    (0): Bottleneck(
      (cv1): Conv(
        (conv): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_stats=True)
        (act): SiLU(inplace=True)
      )
    )
  )
)
```

[104]

```python
# Create dummy input with correct shape
dummy_input = torch.randn(1, 3, 640, 640)
dummy_input = dummy_input.to(device)
# Export to ONNX
torch.onnx.export(
    model,
    dummy_input,
    "yolov5s.onnx",
    opset_version=12,
    # do_constant_folding=False,  # Prevent folding
    input_names=['input'],
    output_names=['output']
)
```

```
/Users/mlfornerds/Projects/yolov5/models/yolo.py:268: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if augment:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:171: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if visualize:
```

```
dummy_input = torch.randn(1, 3, 640, 640)
dummy_input = dummy_input.to(device)
# Export to ONNX
torch.onnx.export(
    model,
    dummy_input,
    "yolov5s.onnx",
    opset_version=12,
    # do_constant_folding=False,  # Prevent folding
    input_names=['input'],
    output_names=['output']
)
```

[104]

```
/Users/mlfornerds/Projects/yolov5/models/yolo.py:268: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if augment:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:171: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if visualize:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:167: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if profile:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:101: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if self.dynamic or self.grid[i].shape[2:4] != x[i].shape[2:4]:
```

```
import onnx
path = "yolov5s
onnx.save(onnx.shape_inference.infer_shapes(onnx.load(path)), path)
```

(module) shape_inference

onnx shape inference. Shape inference is not guaranteed to be complete.

[105]

```
1
```

[ ]

```
!netron yolov5l.onnx --port 8085
```

[5]

```
dummy_input = torch.randn(1, 3, 640, 640)
dummy_input = dummy_input.to(device)
# Export to ONNX
torch.onnx.export(
    model,
    dummy_input,
    "yolov5s.onnx",
    opset_version=12,
    # do_constant_folding=False,  # Prevent folding
    input_names=['input'],
    output_names=['output']
)
```

[104]

```
/Users/mlfornerds/Projects/yolov5/models/yolo.py:268: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if augment:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:171: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if visualize:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:167: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if profile:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:101: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if self.dynamic or self.grid[i].shape[2:4] != x[i].shape[2:4]:
```

```
import onnx
path = "yolov5s.onnx"
onnx.save(onnx.shape_inference.infer_shapes(onnx.load(path)), path)
```

[105]

```
1
```

[ ]

```
!netron yolov5l.onnx --port 8085
```

[5]

```
dummy_input = torch.randn(1, 3, 640, 640)
dummy_input = dummy_input.to(device)
# Export to ONNX
torch.onnx.export(
    (parameter) opset_version: int | None

    opset_version (int, default 17): The version of the
    opset_version=12,
    # do_constant_folding=False,  # Prevent folding
    input_names=['input'],
    output_names=['output']
)
```

[104]

```
/Users/mlfornerds/Projects/yolov5/models/yolo.py:268: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if augment:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:171: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if visualize:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:167: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if profile:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:101: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if self.dynamic or self.grid[i].shape[2:4] != x[i].shape[2:4]:
```

```
import onnx
path = "yolov5s.onnx"
onnx.save(onnx.shape_inference.infer_shapes(onnx.load(path)), path)
```

[105]

```
1
```

[ ]

```
!netron yolov5l.onnx --port 8085
```

[5]

```
    dummy_input,
    "yolov5s.onnx",
    opset_version=12,
    # do_constant_folding=False,  # Prevent folding
    input_names=['input'],
    output_names=['output']
)
```

[104]

```
/Users/mlfornerds/Projects/yolov5/models/yolo.py:268: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if augment:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:171: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if visualize:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:167: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if profile:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:101: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if self.dynamic or self.grid[i].shape[2:4] != x[i].shape[2:4]:
```

```
import onnx
path = "yolov5s.onnx"
onnx.save(onnx.shape_inference.infer_shapes(onnx.load(path)), path)
```

[105]

```
1
```

[ ]

```
!netron yolov5l.onnx --port 8085
```

[5]

```
Serving 'yolov5l.onnx' at http://localhost:8085
^C
Traceback (most recent call last):
  File "/Users/mlfornerds/opt/miniconda3/envs/torch/lib/python3.12/site-packages/netron/server.py", line 265, in wait
    time.sleep(0.1)
```

```
    # do_constant_folding=False,  # Prevent folding
    input_names=['input'],
    output_names=['output']
)
```
[104]

```
/Users/mlfornerds/Projects/yolov5/models/yolo.py:268: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if augment:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:171: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if visualize:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:167: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if profile:
/Users/mlfornerds/Projects/yolov5/models/yolo.py:101: TracerWarning: Converting a tensor to a Python boolean might cause the trace to be incorrect. W
  if self.dynamic or self.grid[i].shape[2:4] != x[i].shape[2:4]:
```

```
import onnx
path = "yolov5s.onnx"
onnx.save(onnx.shape_inference.infer_shapes(onnx.load(path)), path)
```
[105]

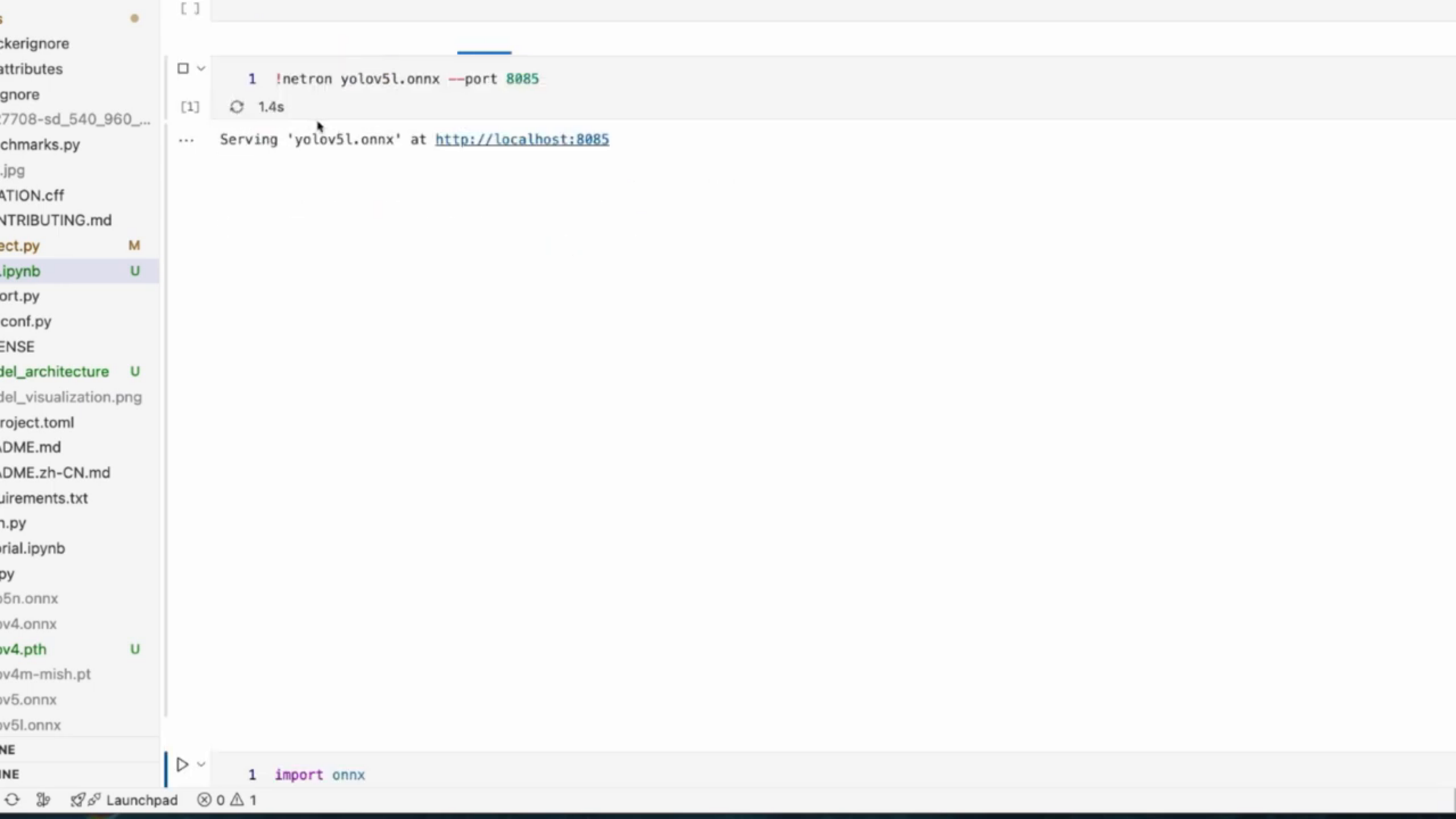
```
1
```
[ ]

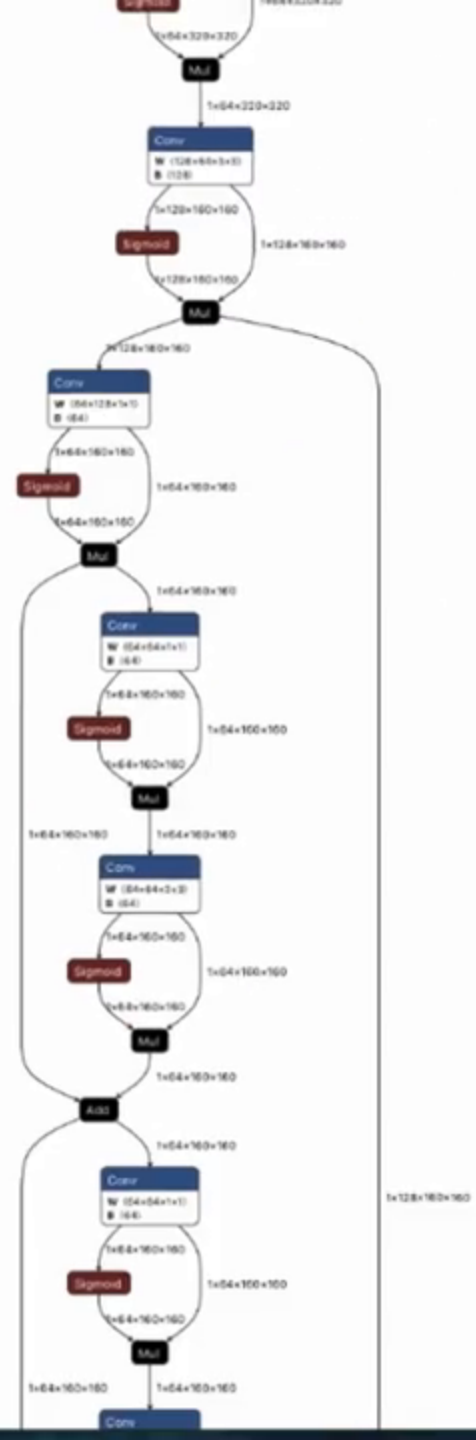
```
!netron yolov5l.onnx --port 8085
```
[5]

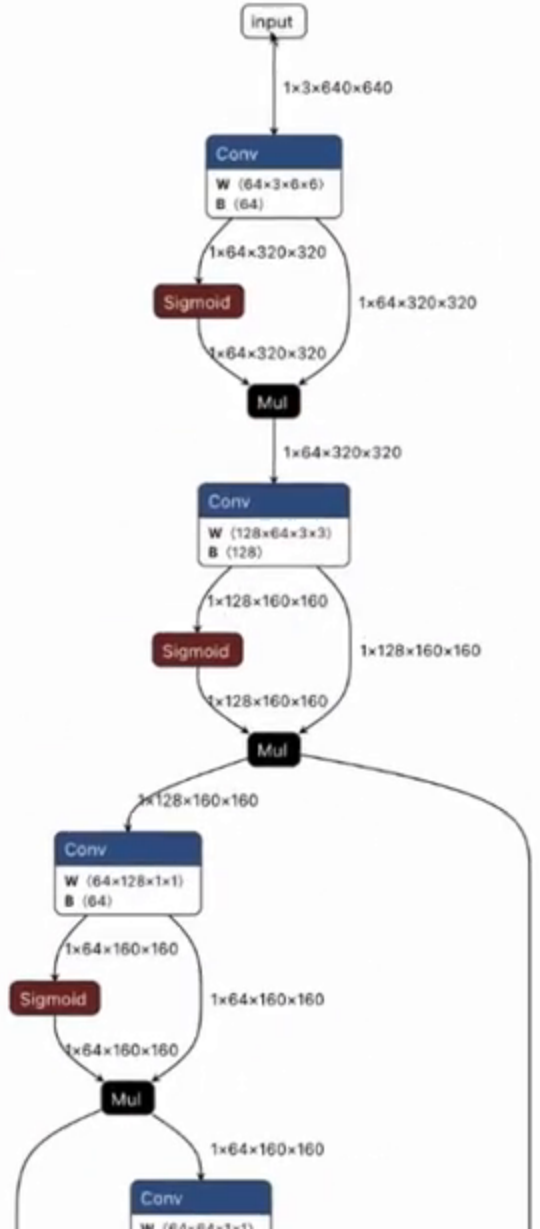
```
Serving 'yolov5l.onnx' at http://localhost:8085
^C
Traceback (most recent call last):
  File "/Users/mlfornerds/opt/miniconda3/envs/torch/lib/python3.12/site-packages/netron/server.py", line 265, in wait
    time.sleep(0.1)
KeyboardInterrupt
```

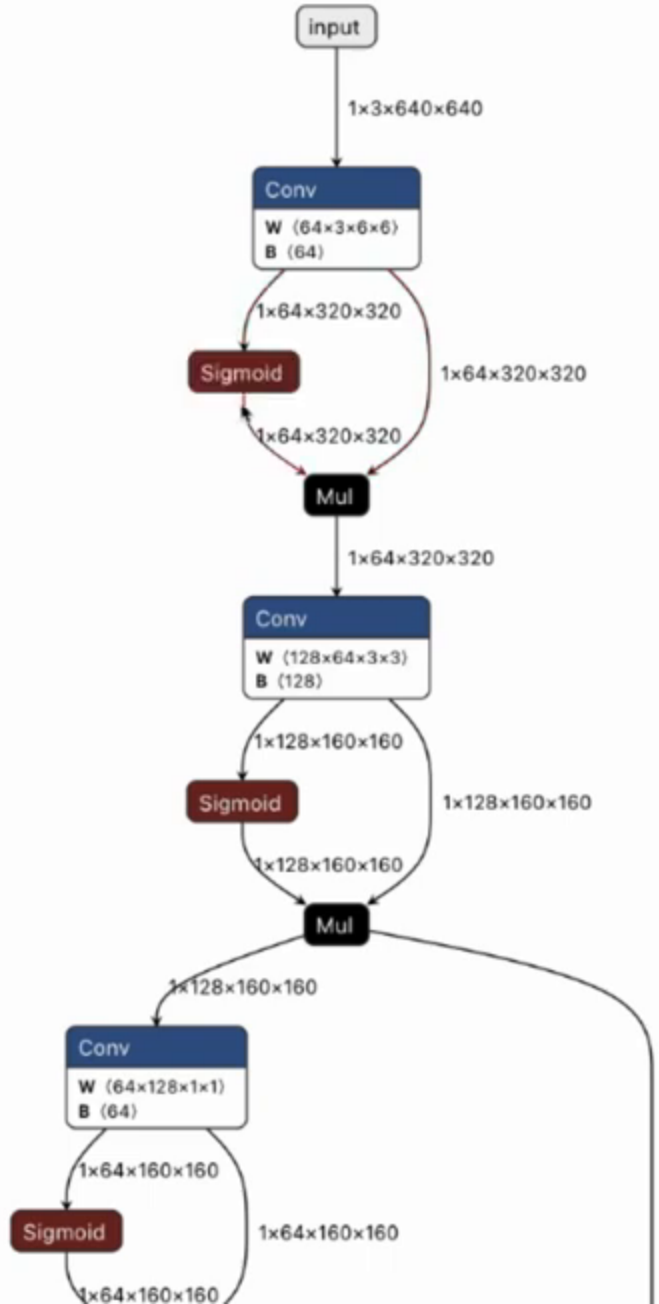
ckerignore
attributes
gnore
7708-sd_540_960_...
chmarks.py
.jpg
ATION.cff
NTRIBUTING.md
ect.py M
.ipynb U
ort.py
conf.py
ENSE
del_architecture U
del_visualization.png
roject.toml
ADME.md
ADME.zh-CN.md
uirements.txt
n.py
rial.ipynb
py
5n.onnx
v4.onnx
v4.pth U
v4m-mish.pt
v5.onnx
v5l.onnx
NE
INE

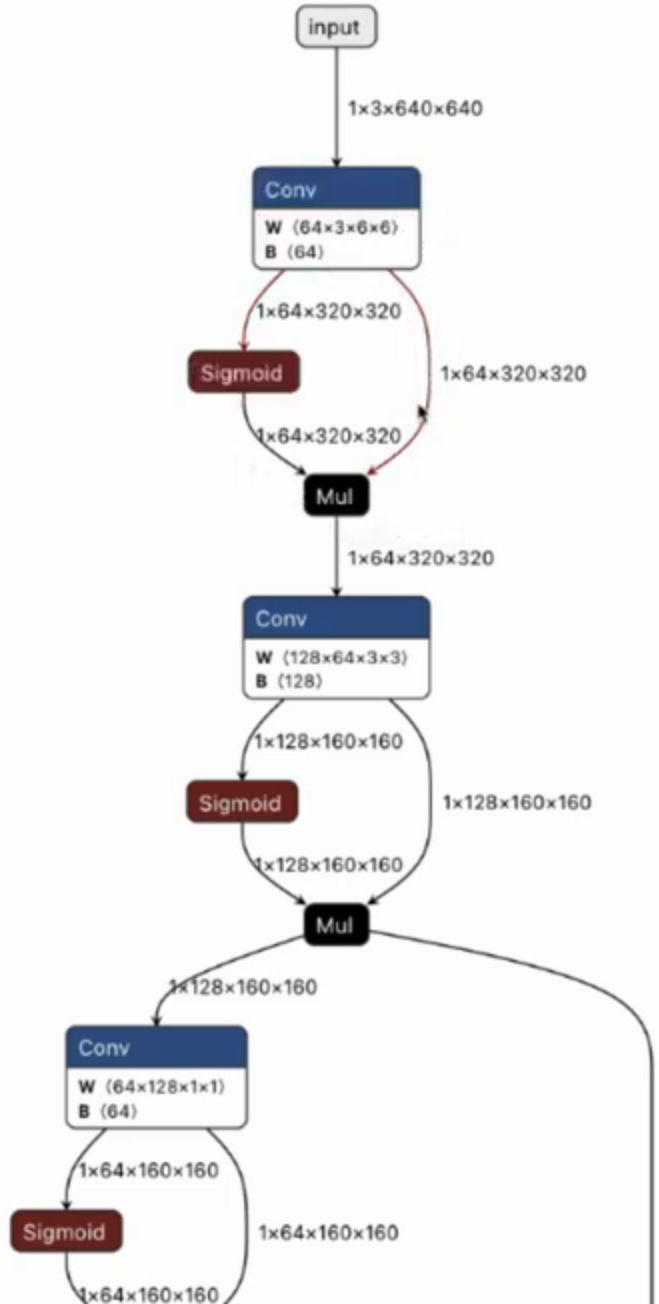
Launchpad ⊗ 0 ⚠ 1

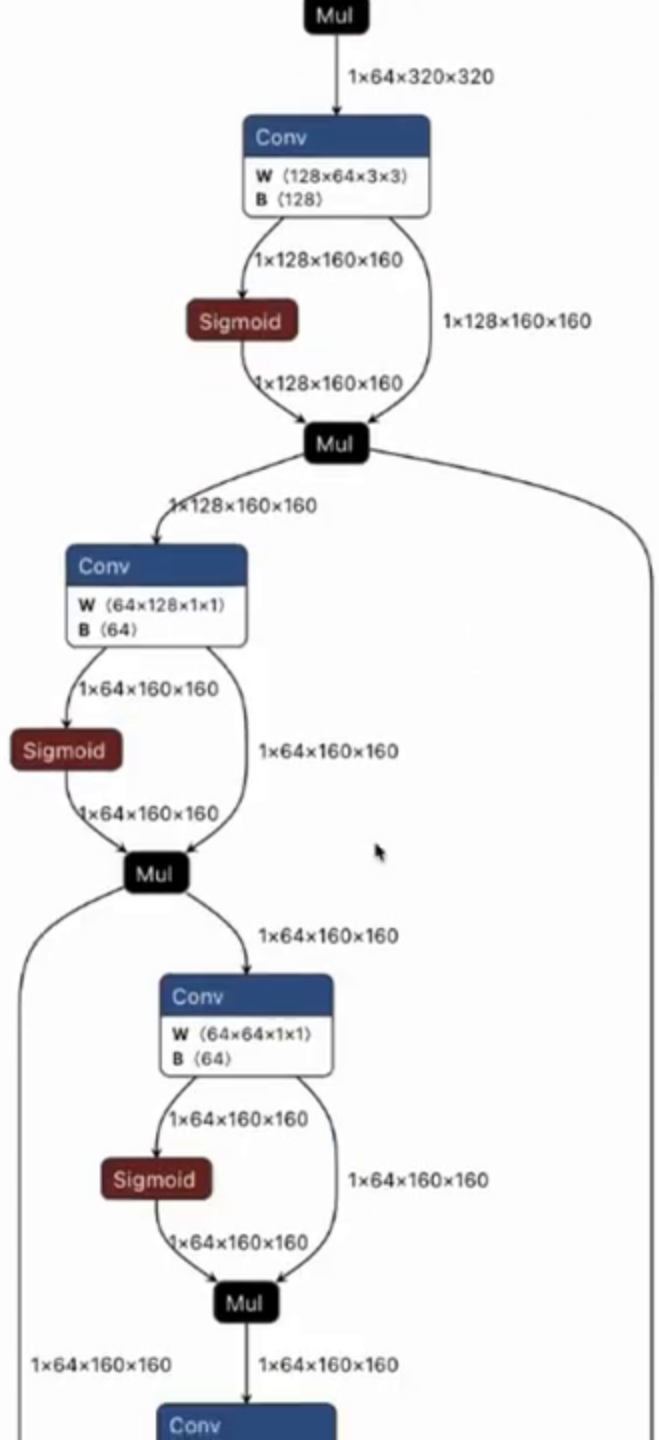

[ ]
1 !netron yolov5l.onnx --port 8085
[1] 1.4s
Serving 'yolov5l.onnx' at http://localhost:8085
1 import onnx
Launchpad
0 1

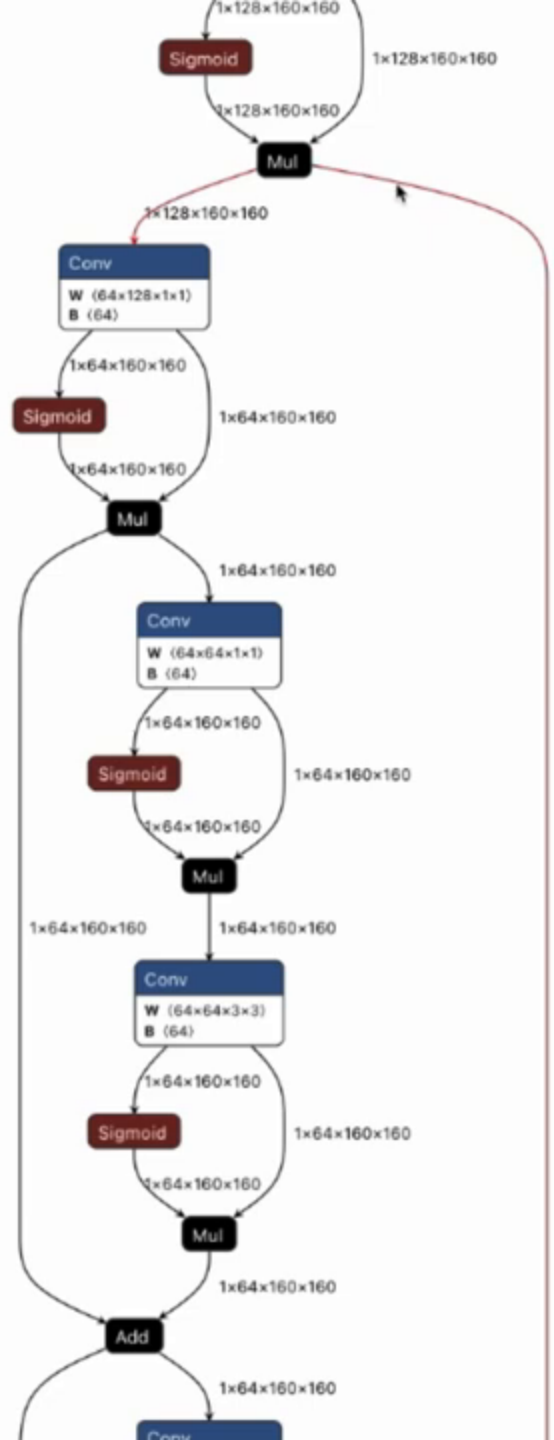

Mul
1x64x220x220
Conv
W (128x64x3x3)
B (128)
1x128x160x160
Sigmoid
1x128x160x160
Mul
1x128x160x160
Conv
W (64x128x1x1)
B (64)
1x64x160x160
Sigmoid
1x64x160x160
Mul
1x64x160x160
Conv
W (64x64x1x1)
B (64)
1x64x160x160
Sigmoid
1x64x160x160
Mul
1x64x160x160
1x64x160x160
Conv
W (64x64x3x3)
B (64)
1x64x160x160
Sigmoid
1x64x160x160
Mul
1x64x160x160
Add
1x64x160x160
Conv
W (64x64x1x1)
B (64)
1x64x160x160
Sigmoid
1x64x160x160
Mul
1x64x160x160
1x64x160x160
Conv
1x128x160x160

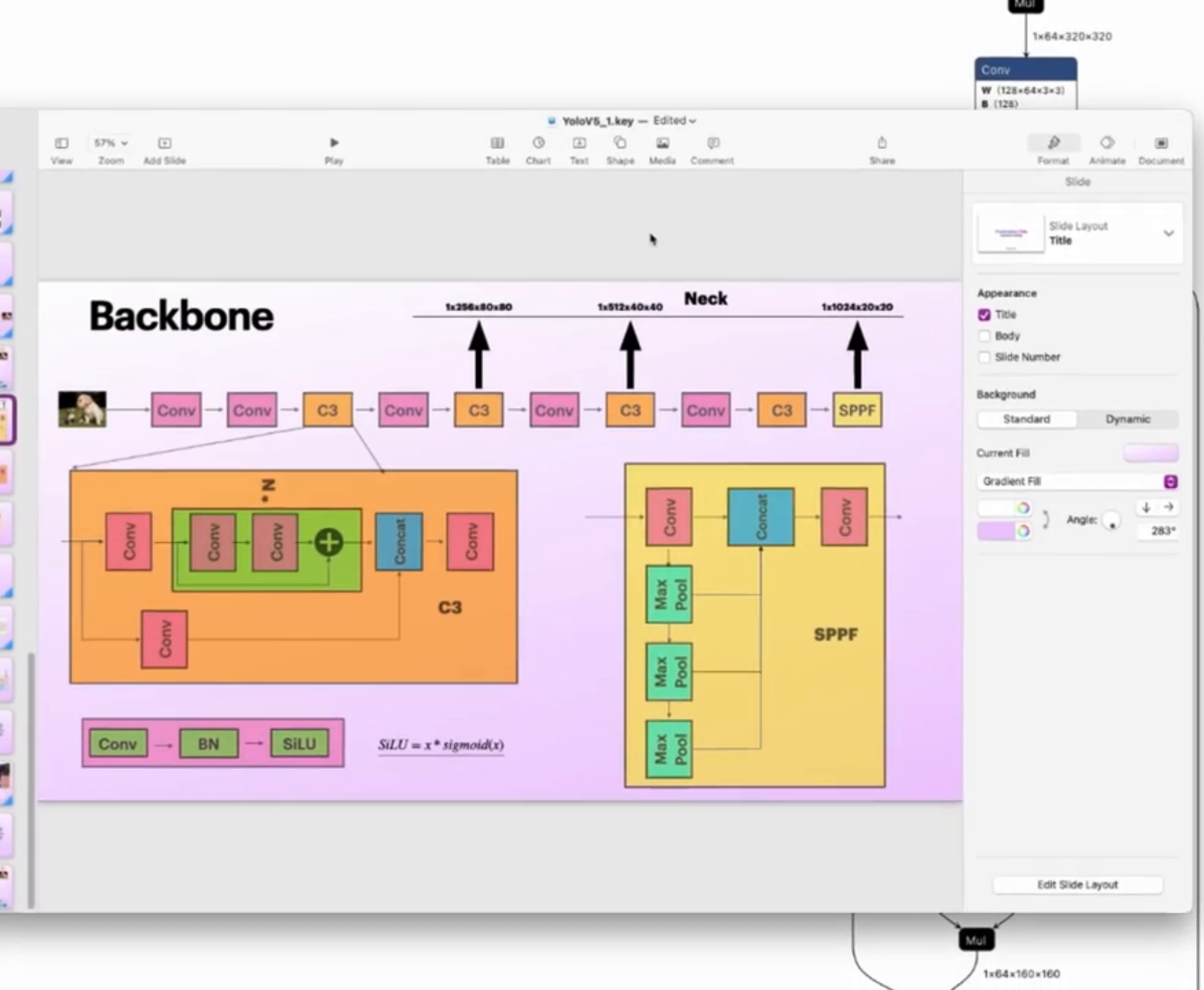

input
1×3×640×640
Conv
W (64×3×6×6)
B (64)
1×64×320×320
Sigmoid
1×64×320×320
1×64×320×320
Mul
1×64×320×320
Conv
W (128×64×3×3)
B (128)
1×128×160×160
Sigmoid
1×128×160×160
1×128×160×160
Mul
1×128×160×160
Conv
W (64×128×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×64×160×160
Conv


input
1×3×640×640
Conv
W (64×3×6×6)
B (64)
1×64×320×320
Sigmoid
1×64×320×320
1×64×320×320
Mul
1×64×320×320
Conv
W (128×64×3×3)
B (128)
1×128×160×160
Sigmoid
1×128×160×160
1×128×160×160
Mul
1×128×160×160
Conv
W (64×128×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160


input
1×3×640×640
Conv
W (64×3×6×6)
B (64)
1×64×320×320
Sigmoid
1×64×320×320
1×64×320×320
Mul
1×64×320×320
Conv
W (128×64×3×3)
B (128)
1×128×160×160
Sigmoid
1×128×160×160
1×128×160×160
Mul
1×128×160×160
Conv
W (64×128×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160

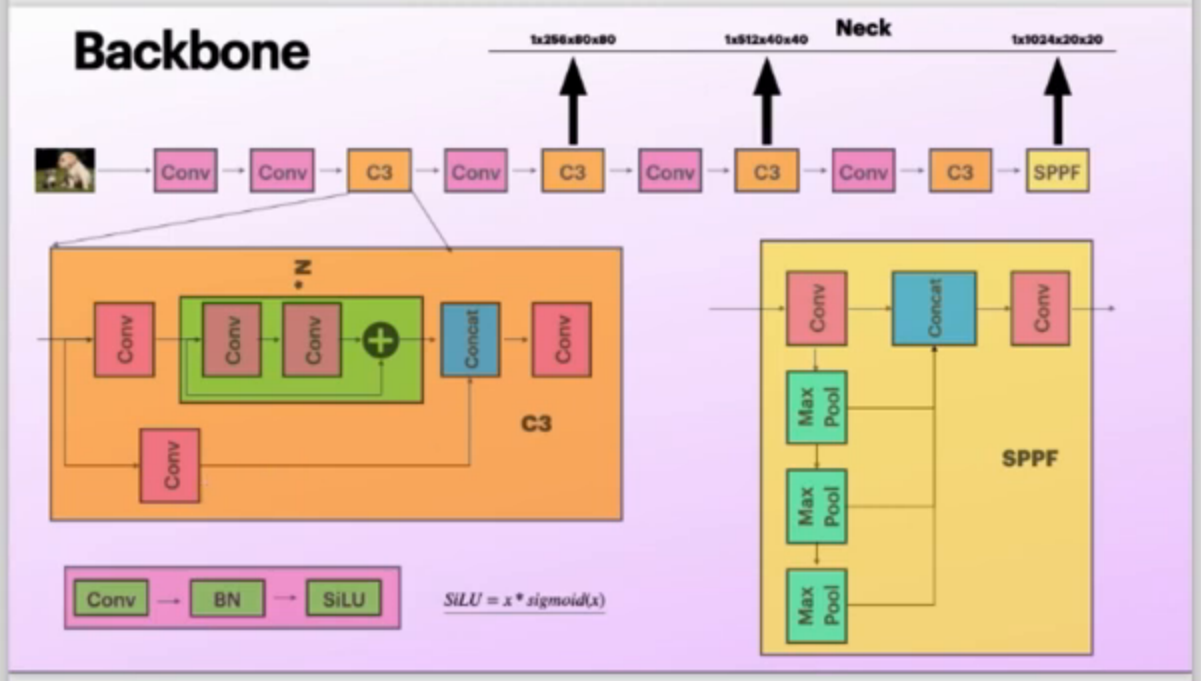

Mul
1×64×320×320
Conv
W (128×64×3×3)
B (128)
1×128×160×160
Sigmoid
1×128×160×160
1×128×160×160
Mul
1×128×160×160
Conv
W (64×128×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×64×160×160
Conv
W (64×64×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×64×160×160
1×64×160×160
Conv

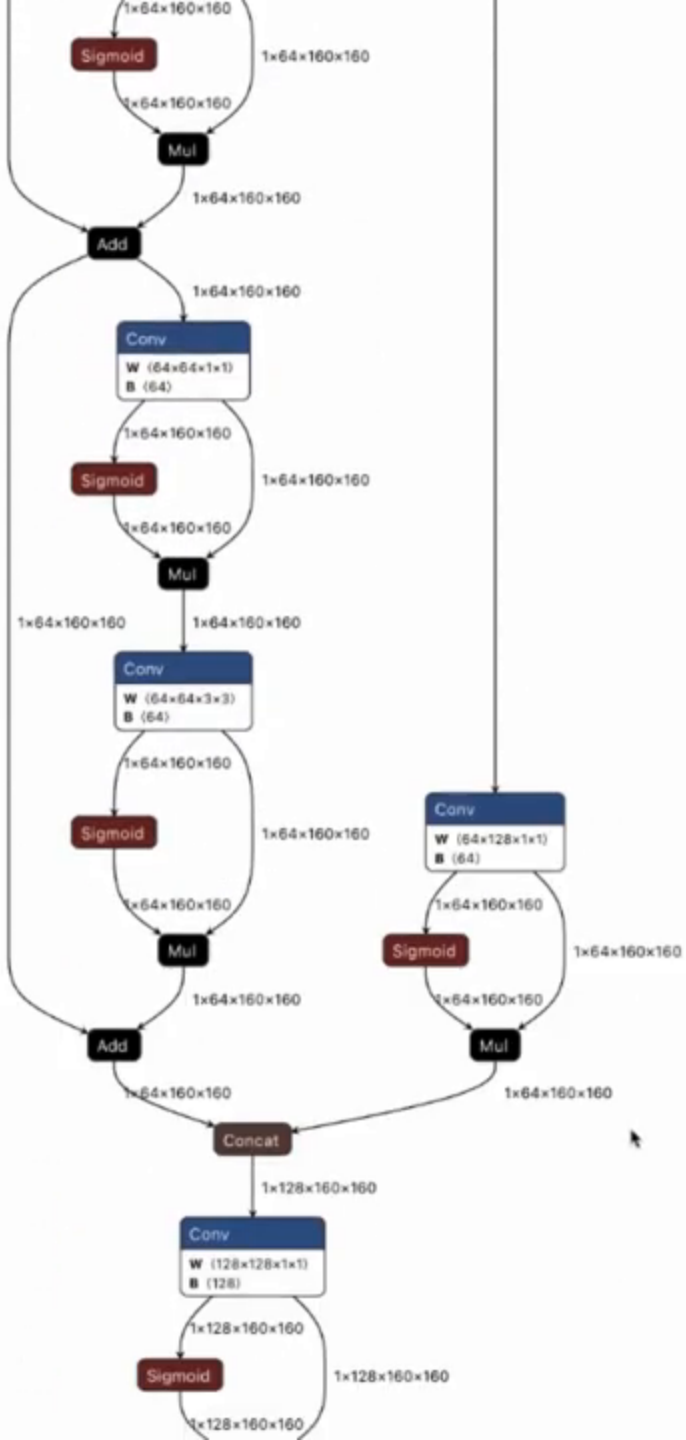

1×128×160×160
Sigmoid
1×128×160×160
1×128×160×160
Mul
1×128×160×160
Conv
W (64×128×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×64×160×160
Conv
W (64×64×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×64×160×160
1×64×160×160
Conv
W (64×64×3×3)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×64×160×160
Add
1×64×160×160
Conv


Mul
1×64×320×320
Conv
W (128×64×3×3)
B (128)
YoloV5_1.key — Edited
View
Zoom
Add Slide
Play
Table
Chart
Text
Shape
Media
Comment
Share
Format
Animate
Document
Slide
Slide Layout
Title
Appearance
Title
Body
Slide Number
Background
Standard
Dynamic
Current Fill
Gradient Fill
Angle:
283°
Edit Slide Layout
Backbone
Neck
1x256x80x80
1x512x40x40
1x1024x20x20
Conv
Conv
C3
Conv
C3
Conv
C3
Conv
C3
SPPF
N
Conv
Conv
Conv
Concat
Conv
Conv
C3
Conv
Concat
Conv
Max Pool
Max Pool
Max Pool
SPPF
Conv
BN
SiLU
SiLU = x * sigmoid(x)
Mul
1×64×160×160

YoloV5_1.key — Edited

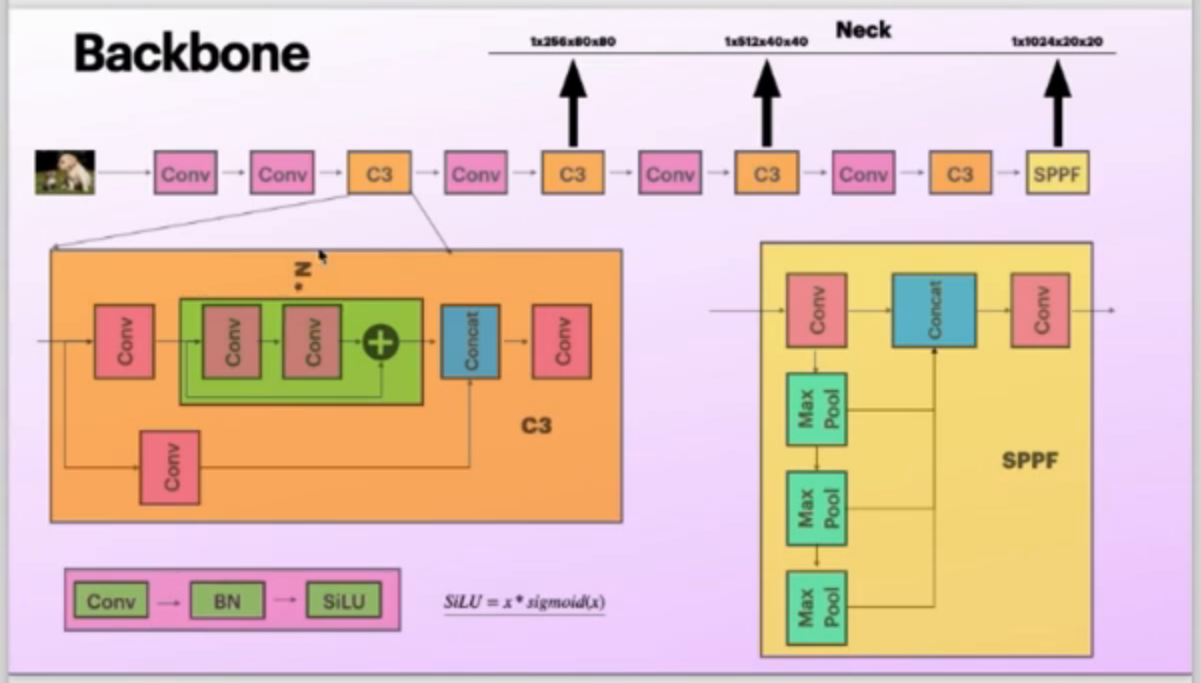
View Zoom Add Slide Play Table Chart Text Shape Media Comment Share Format Animate Document

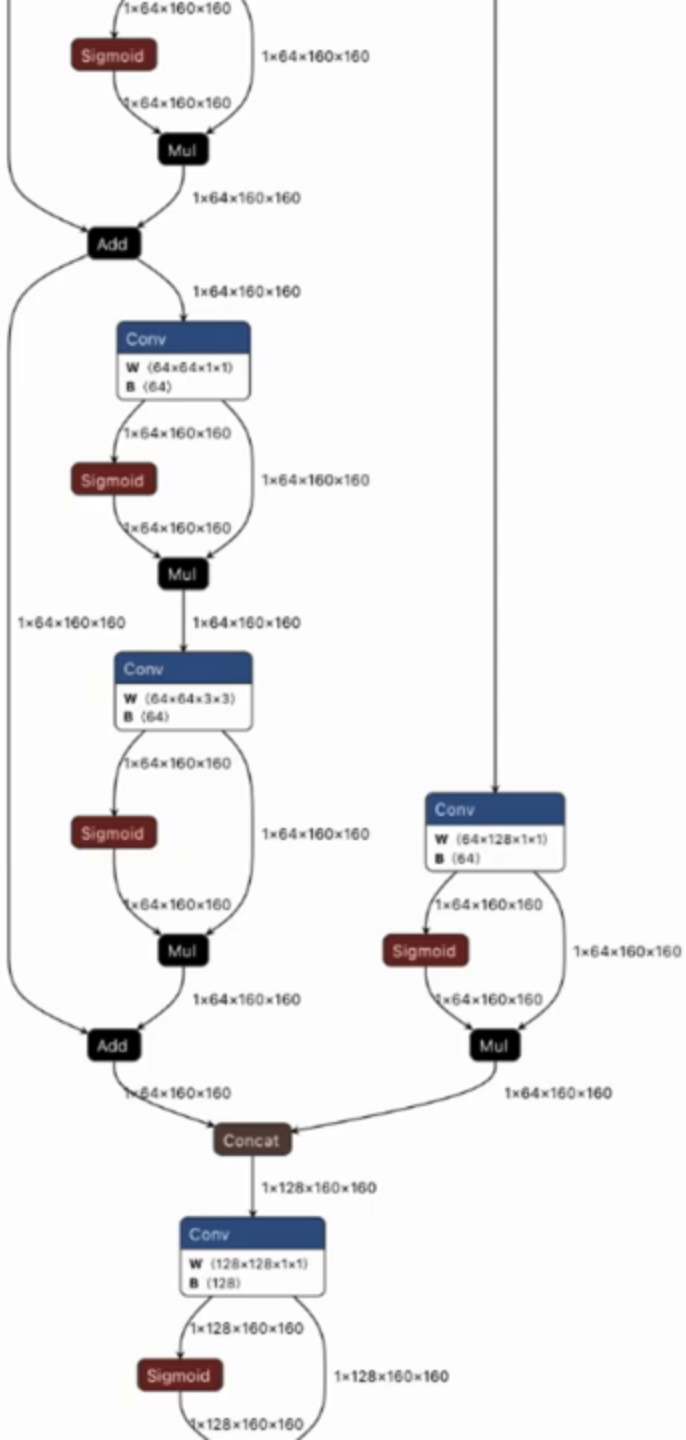


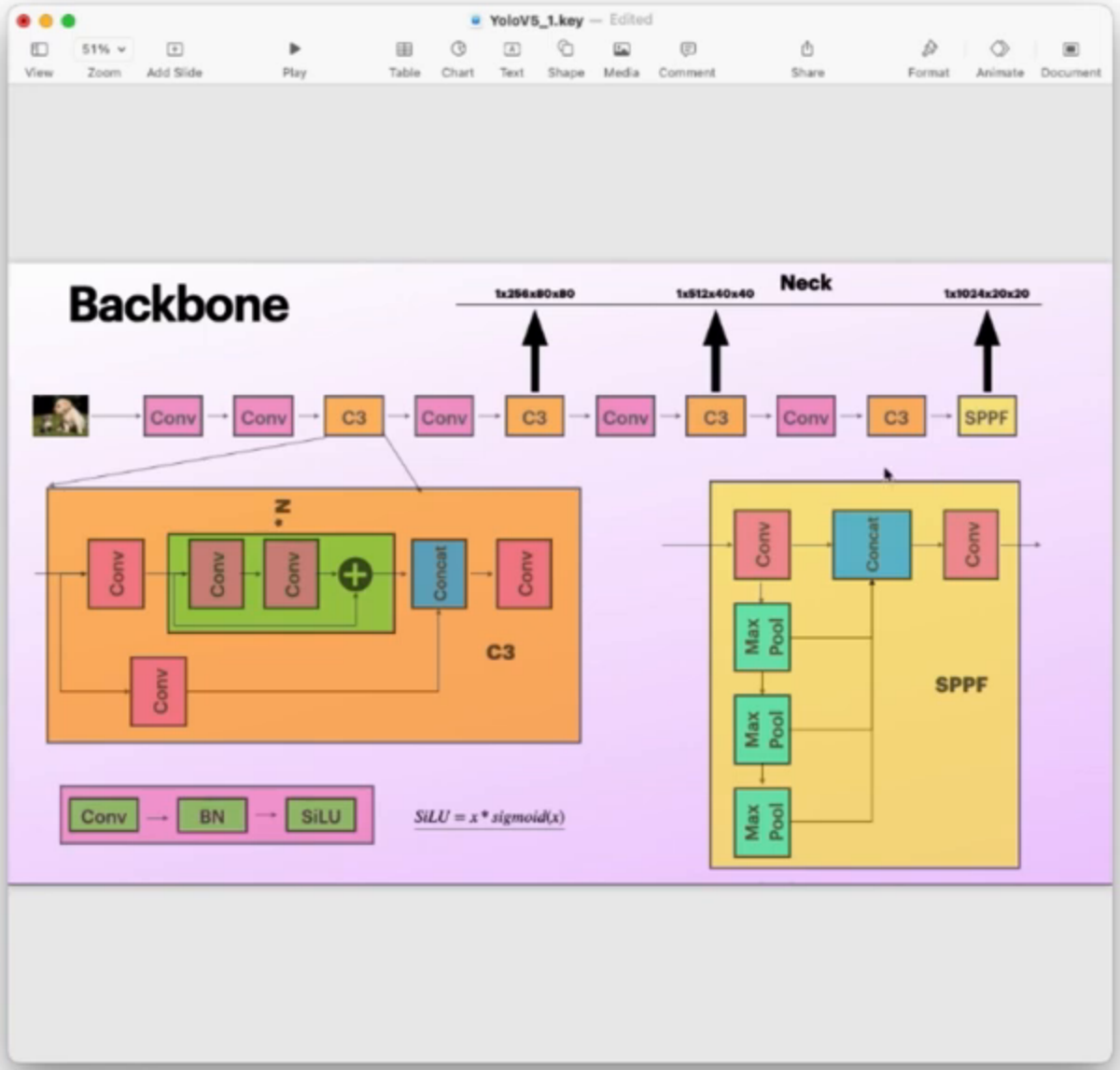


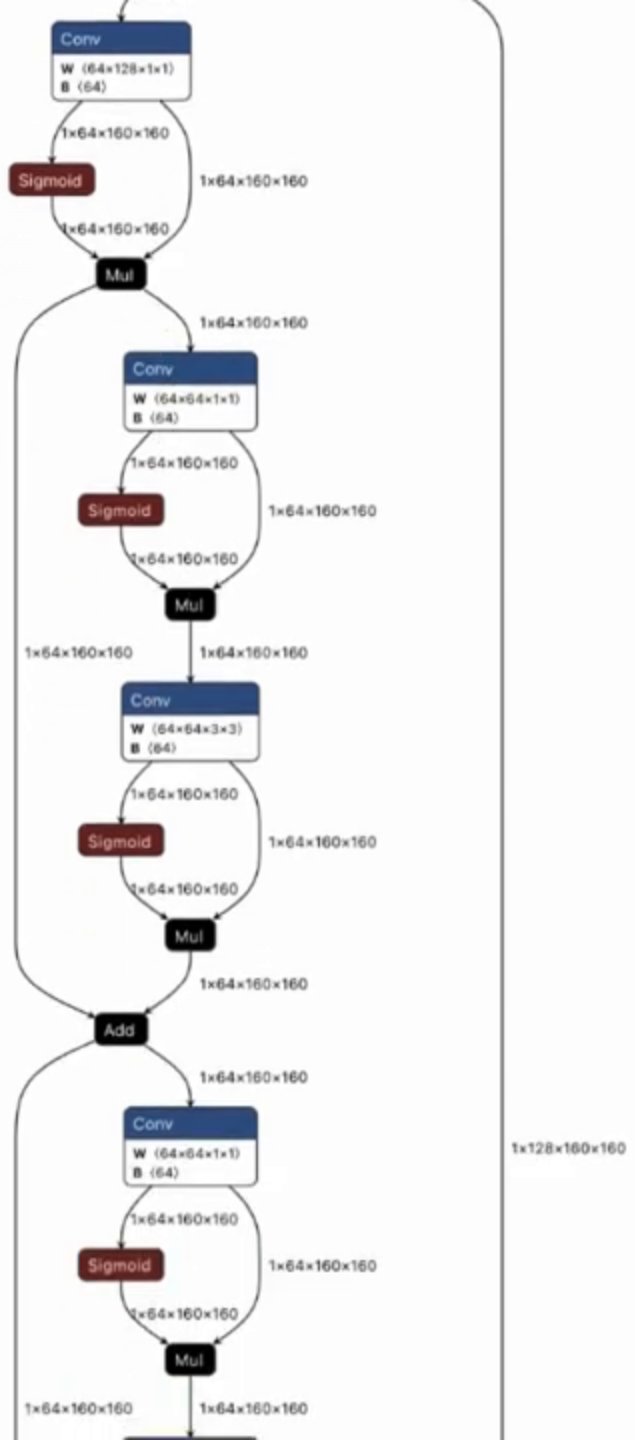
YoloV5_1.key — Edited

View Zoom Add Slide Play Table Chart Text Shape Media Comment Share Format Animate Document



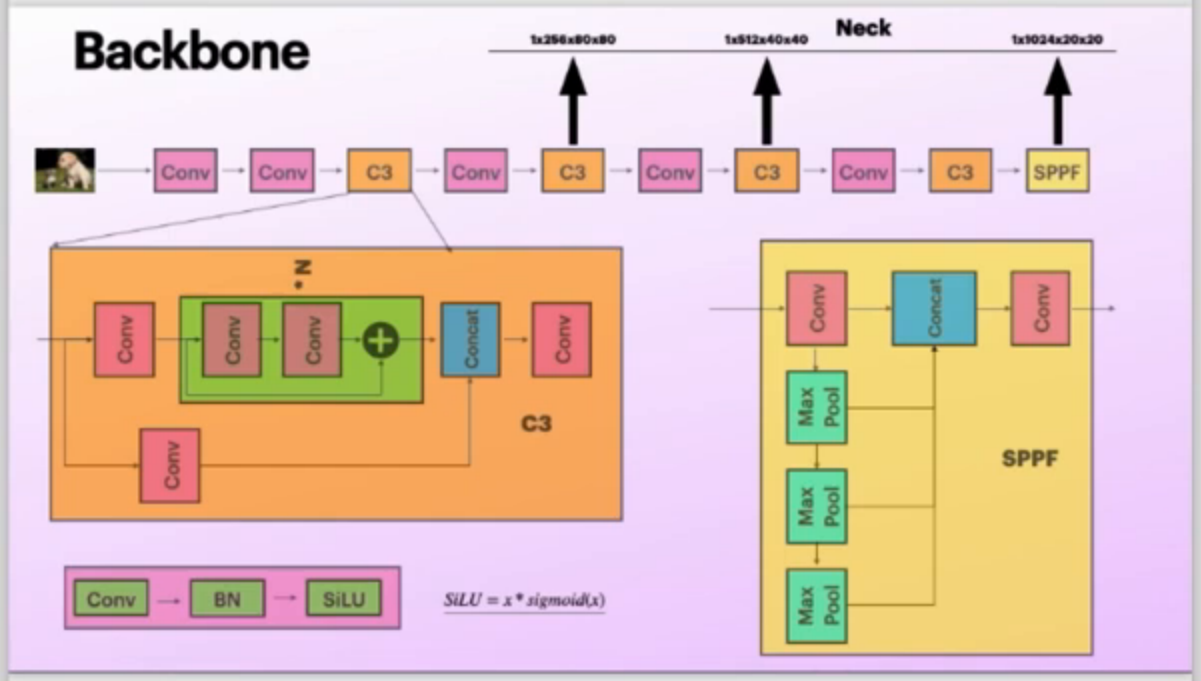


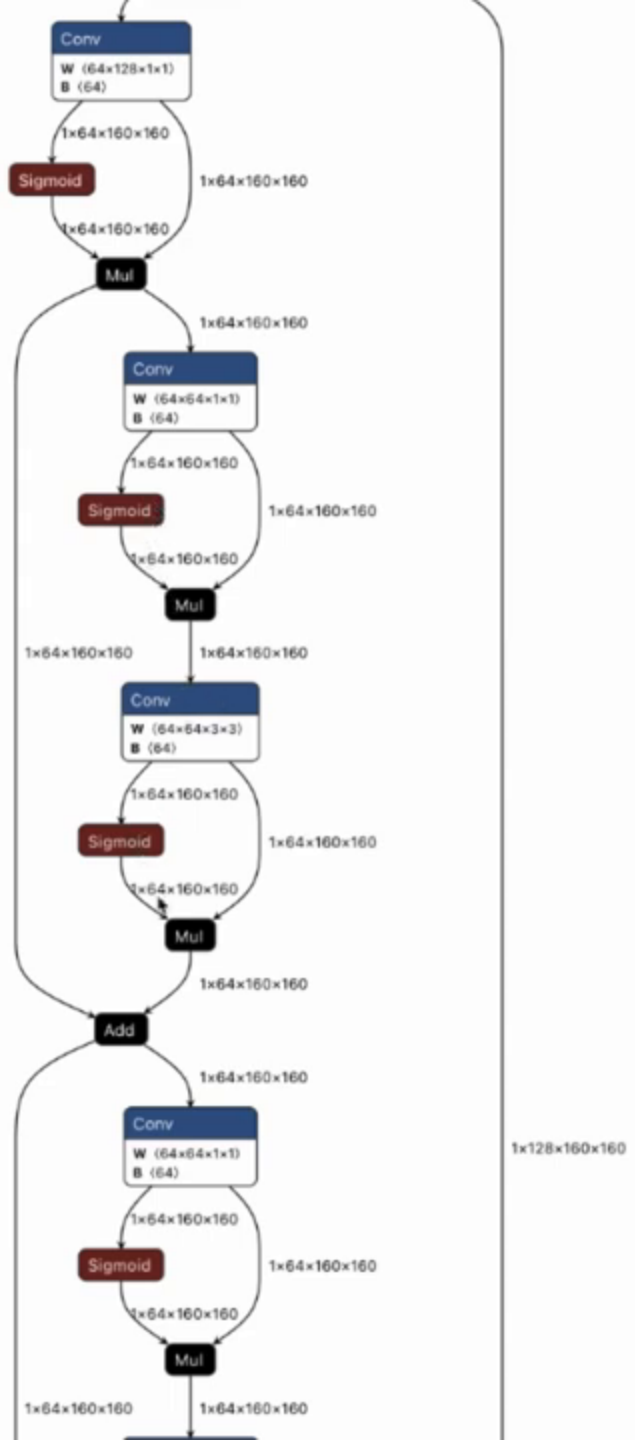

YoloV5_1.key — Edited
51%
View
Zoom
Add Slide
Play
Table
Chart
Text
Shape
Media
Comment
Share
Format
Animate
Document
Backbone
Neck
1x256x80x80
1x512x40x40
1x1024x20x20
Conv
Conv
C3
Conv
C3
Conv
C3
Conv
C3
SPPF
N
Conv
Conv
Conv
Concat
Conv
Conv
C3
Conv
Concat
Conv
Max Pool
Max Pool
Max Pool
SPPF
Conv
BN
SiLU
SiLU = x * sigmoid(x)

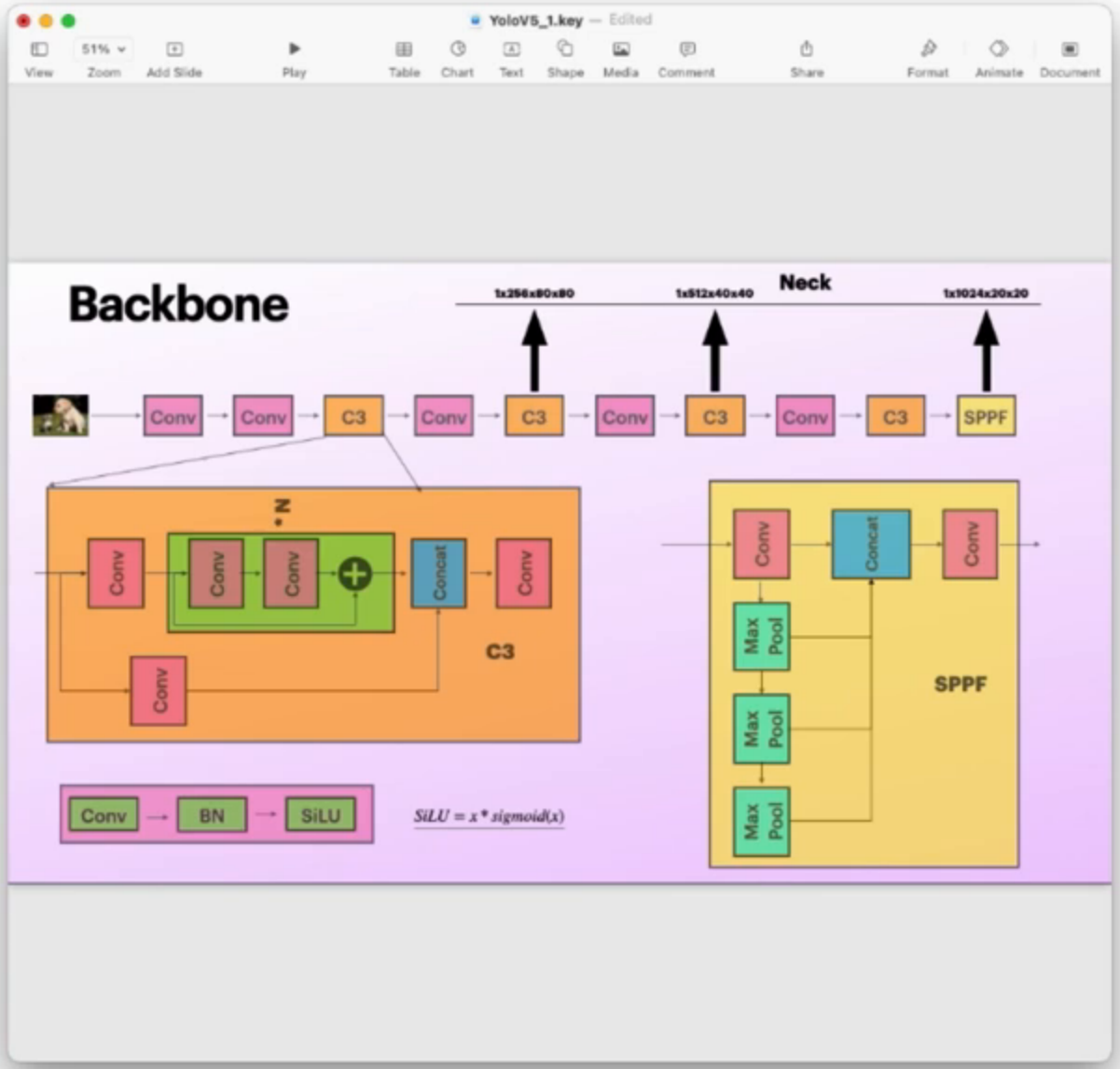

Conv
W (64×128×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×64×160×160
Conv
W (64×64×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×64×160×160
1×64×160×160
Conv
W (64×64×3×3)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×64×160×160
Add
1×64×160×160
Conv
W (64×64×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×64×160×160
1×64×160×160
1×128×160×160

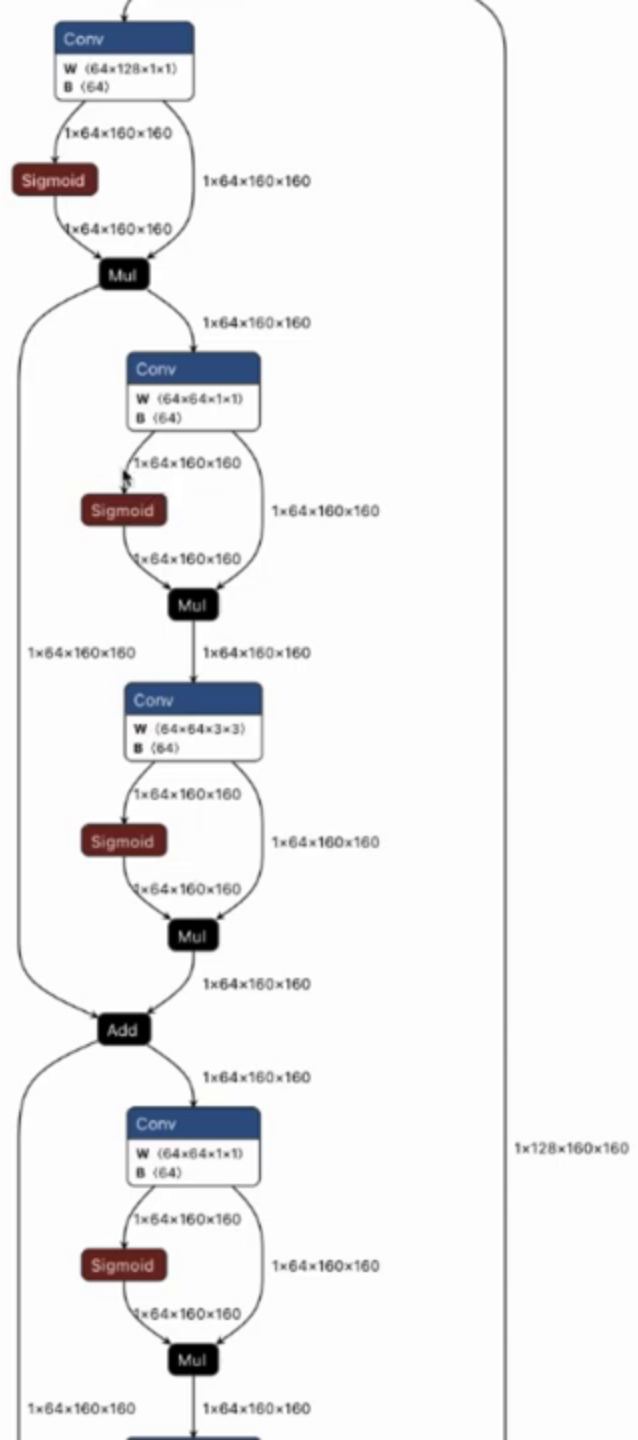
YoloV5_1.key — Edited

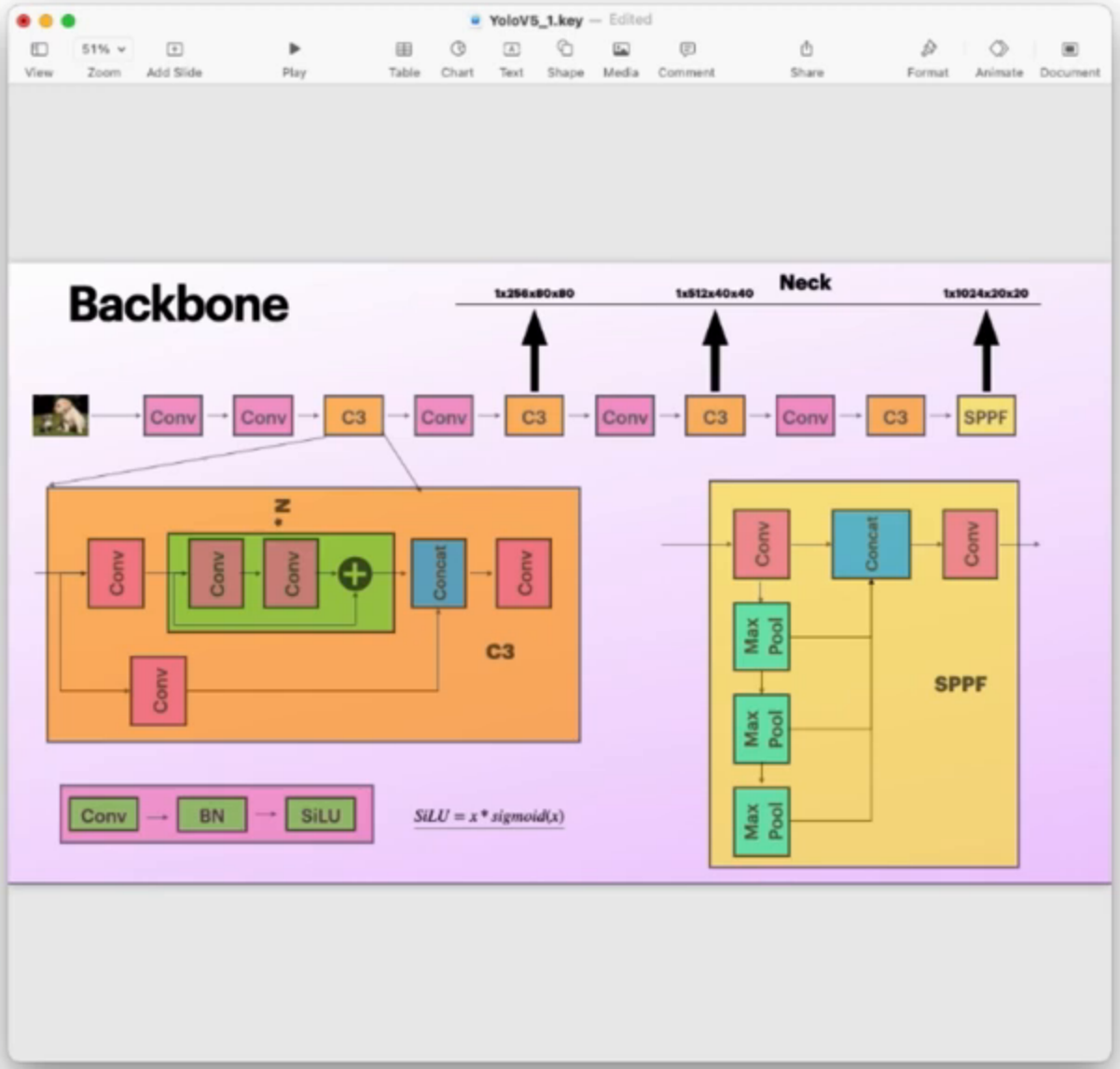
View Zoom Add Slide Play Table Chart Text Shape Media Comment Share Format Animate Document

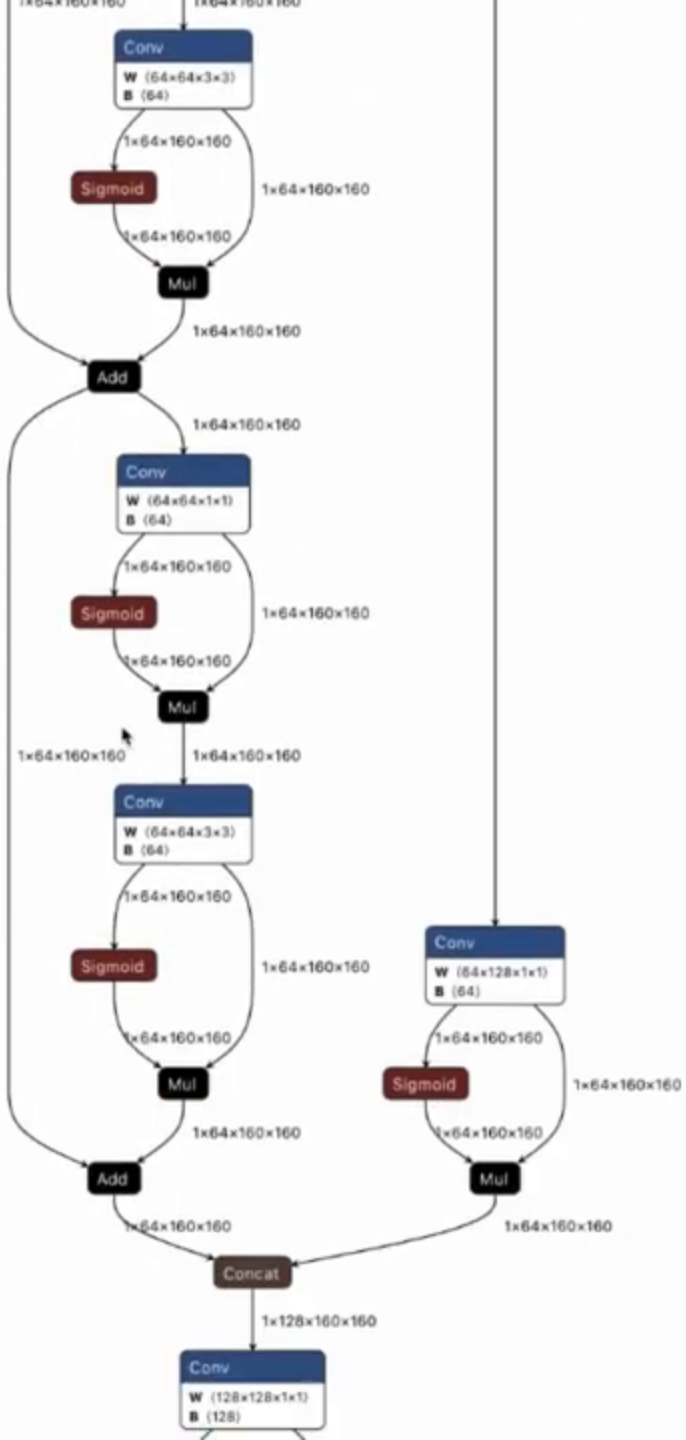


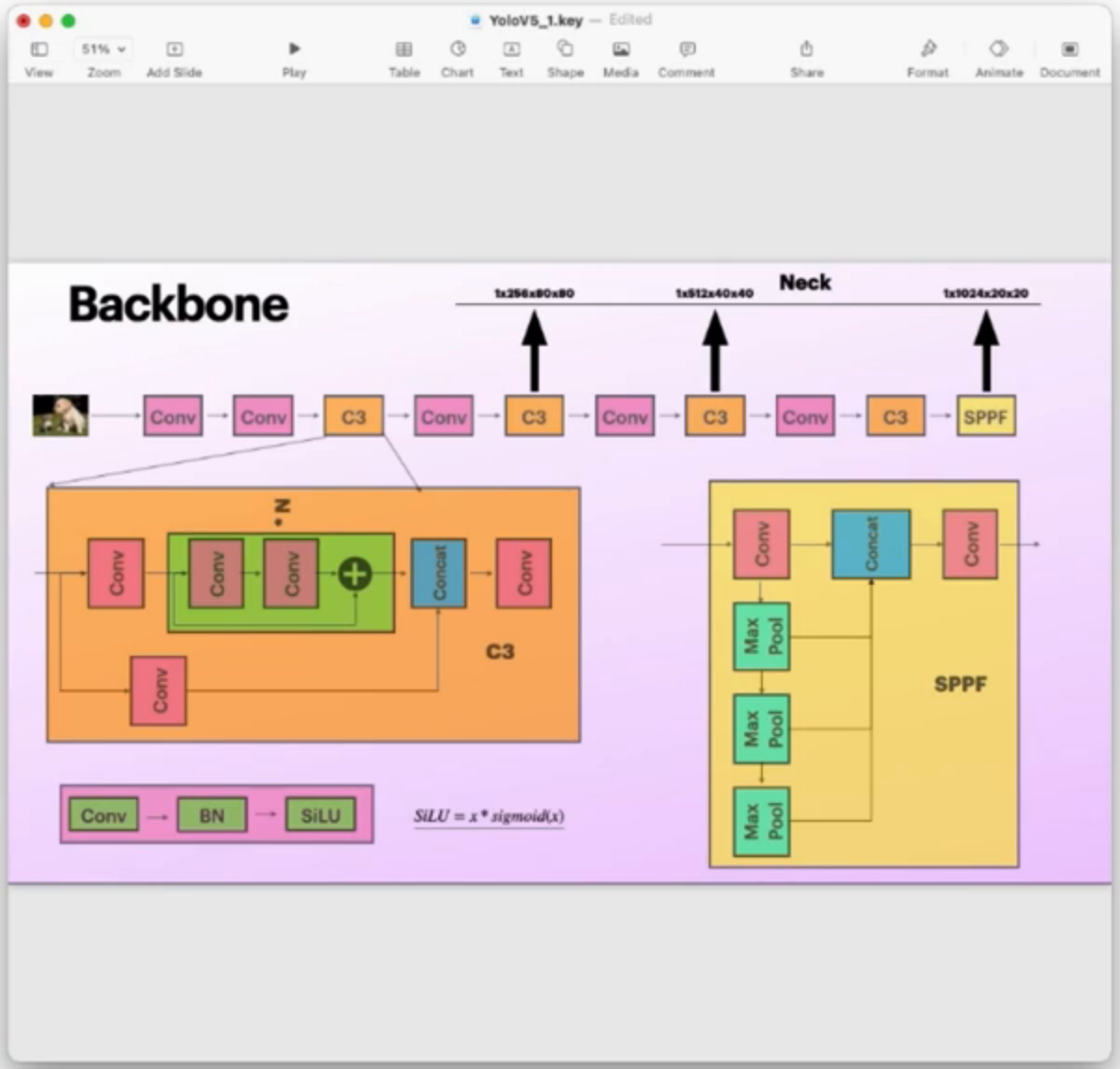


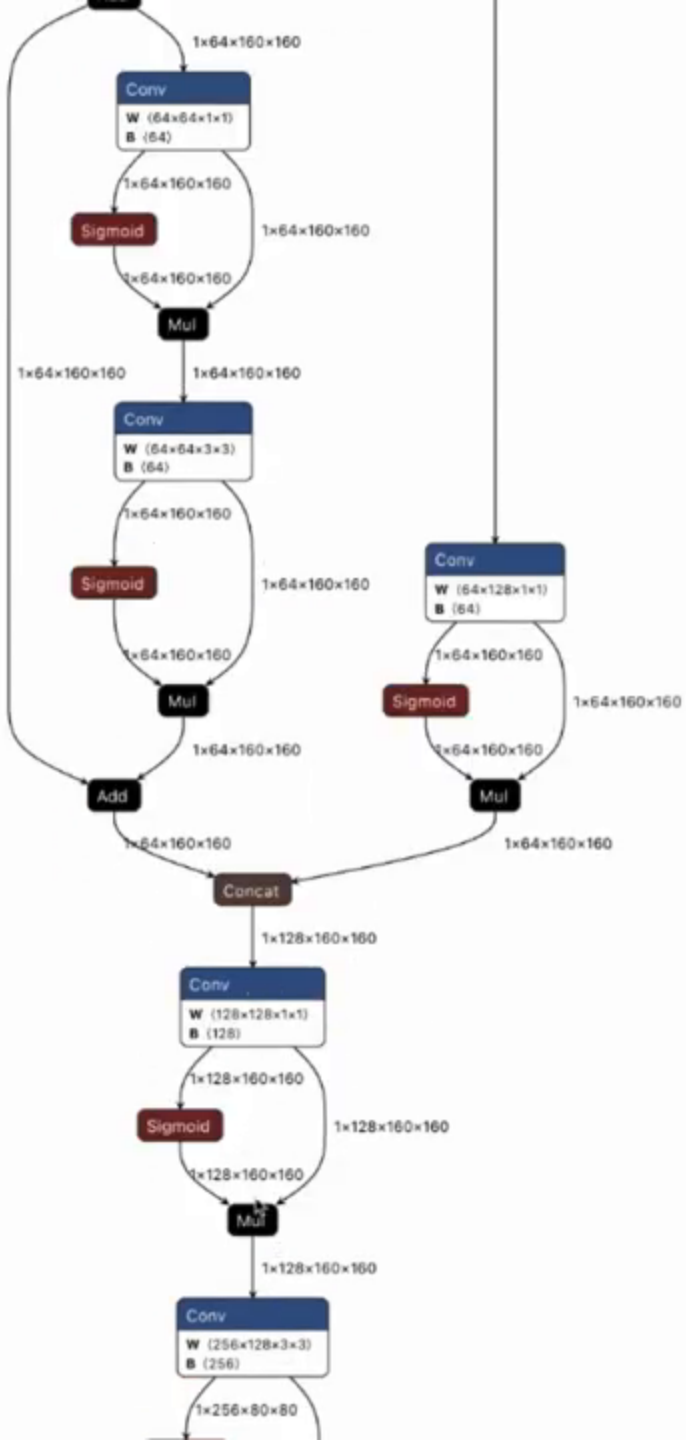

YoloV5_1.key — Edited
View
Zoom
Add Slide
Play
Table
Chart
Text
Shape
Media
Comment
Share
Format
Animate
Document
Backbone
Neck
1x256x80x80
1x512x40x40
1x1024x20x20
Conv
C3
SPPF
Concat
Max Pool
BN
SiLU
SiLU = x * sigmoid(x)

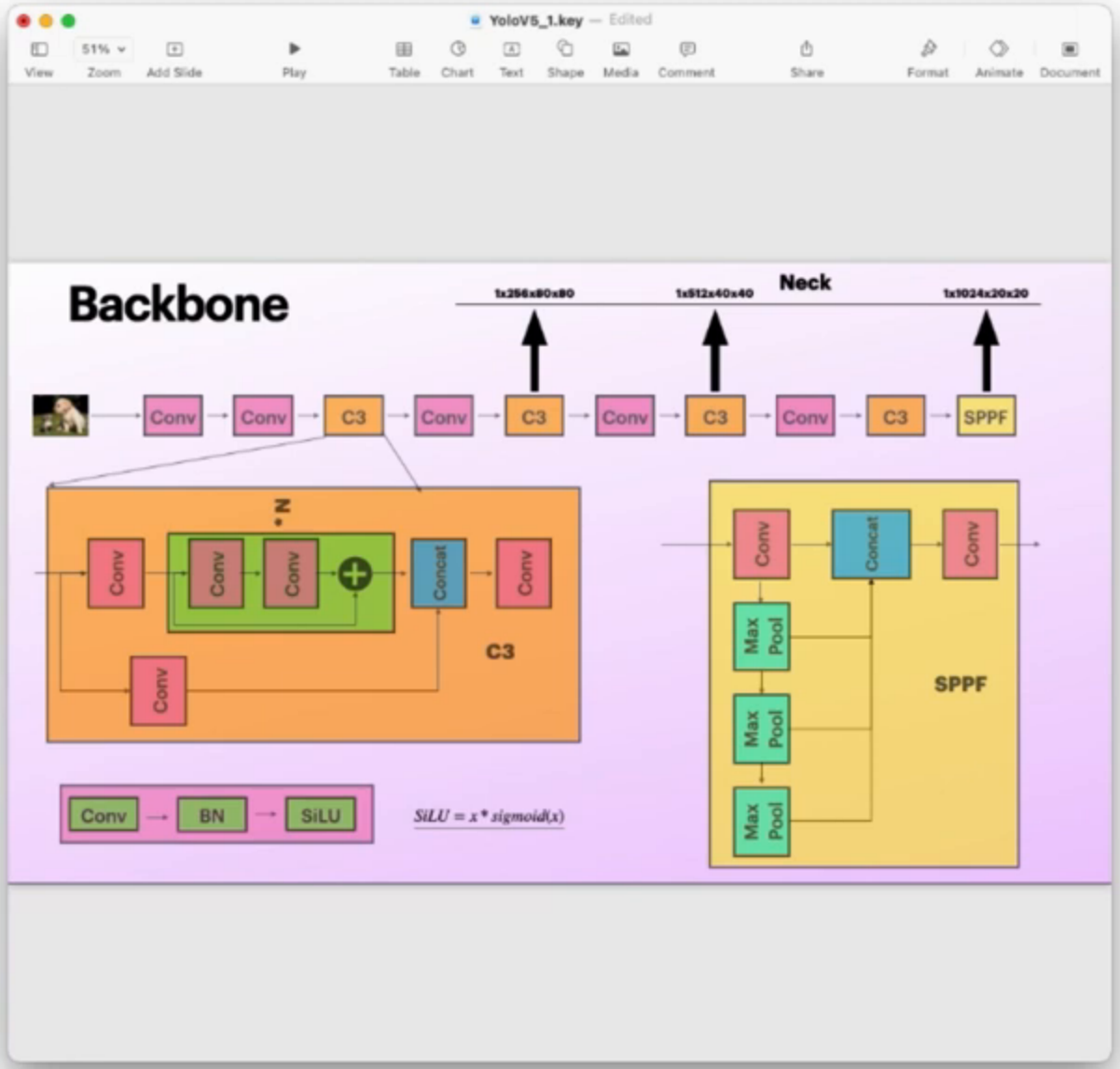

Conv
W (64×128×1×1)
B (64)
1×64×160×160
Sigmoid
Mul
Conv
W (64×64×1×1)
B (64)
Conv
W (64×64×3×3)
B (64)
Add
1×128×160×160

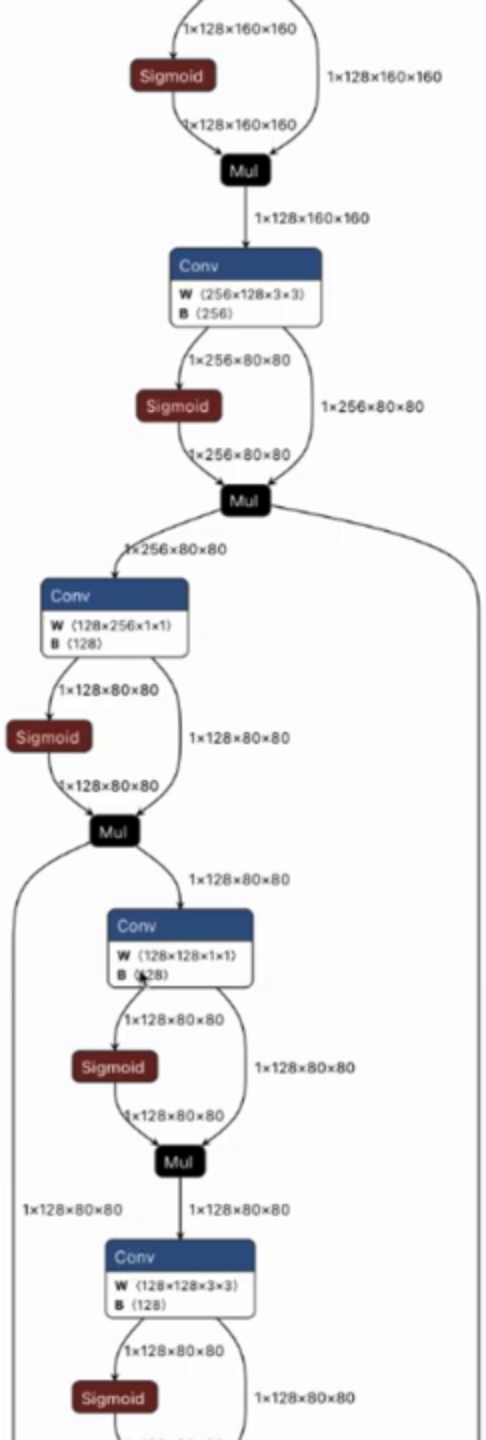

YoloV5_1.key — Edited
View
Zoom
Add Slide
Play
Table
Chart
Text
Shape
Media
Comment
Share
Format
Animate
Document
Backbone
Neck
1x256x80x80
1x512x40x40
1x1024x20x20
Conv
C3
SPPF
Concat
Max Pool
BN
SiLU
SiLU = x * sigmoid(x)

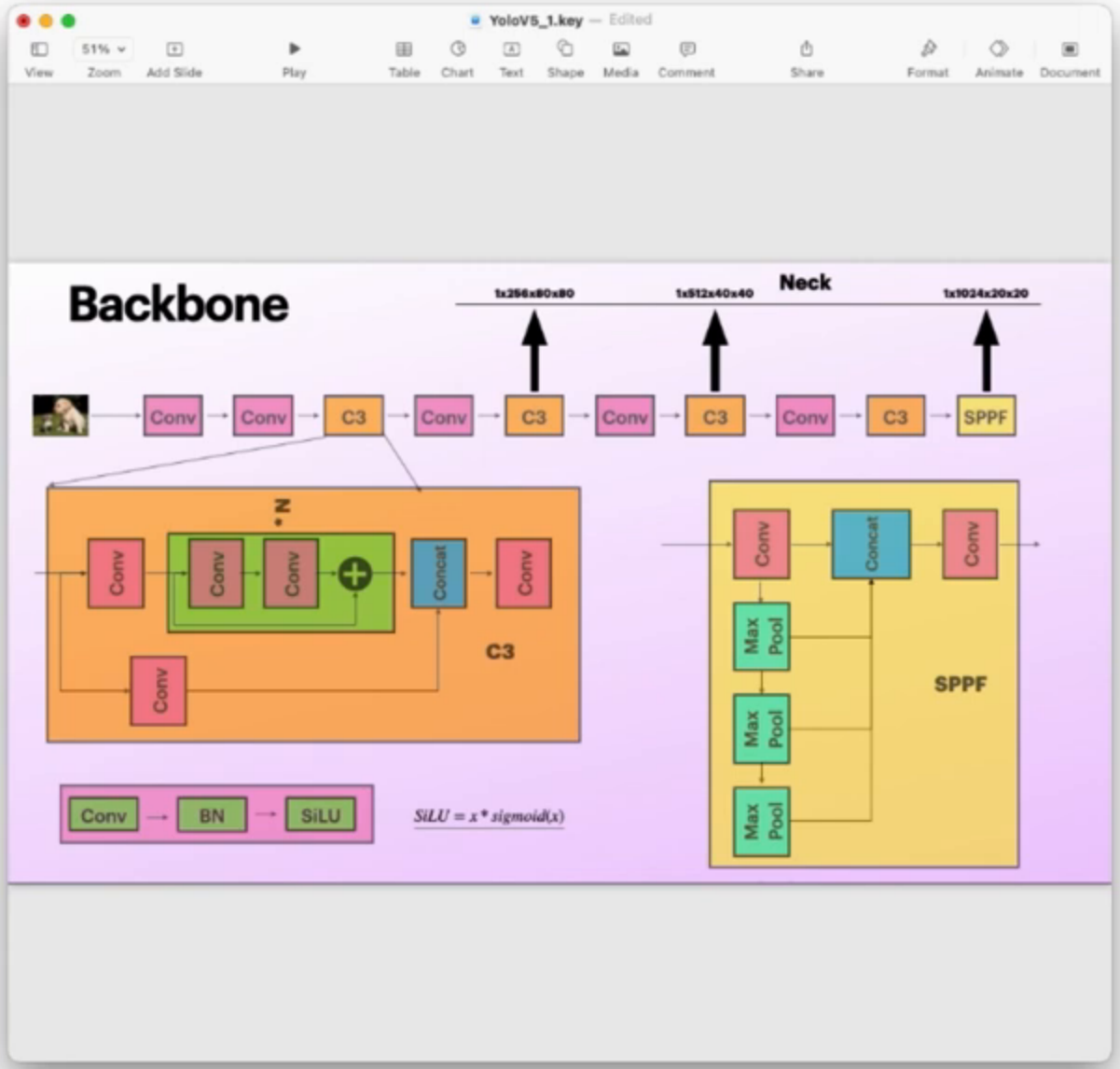

Conv
W (64×64×3×3)
B (64)
1×64×160×160
Sigmoid
Mul
Add
W (64×64×1×1)
W (64×128×1×1)
B (64)
W (128×128×1×1)
B (128)
Concat
1×128×160×160

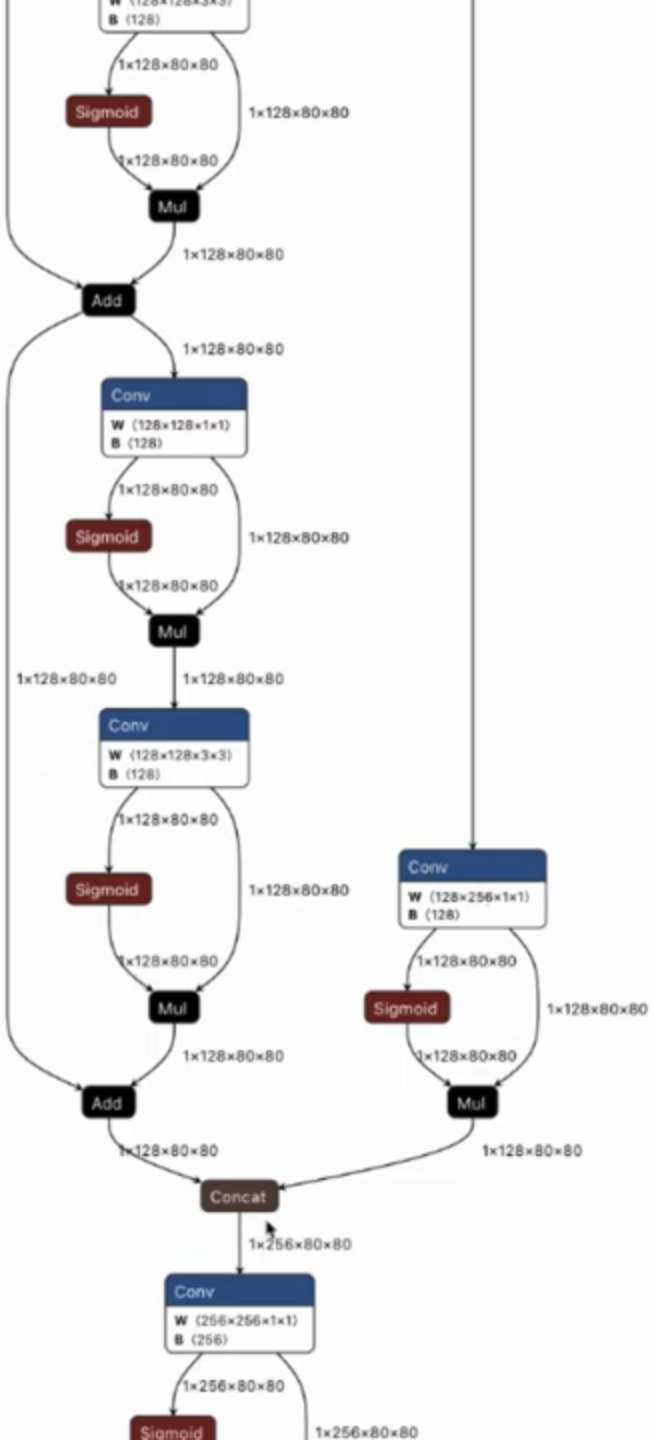

YoloV5_1.key — Edited
Backbone
Neck
1x256x80x80
1x512x40x40
1x1024x20x20
Conv
C3
SPPF
N
Concat
Max Pool
Conv → BN → SiLU
SiLU = x * sigmoid(x)

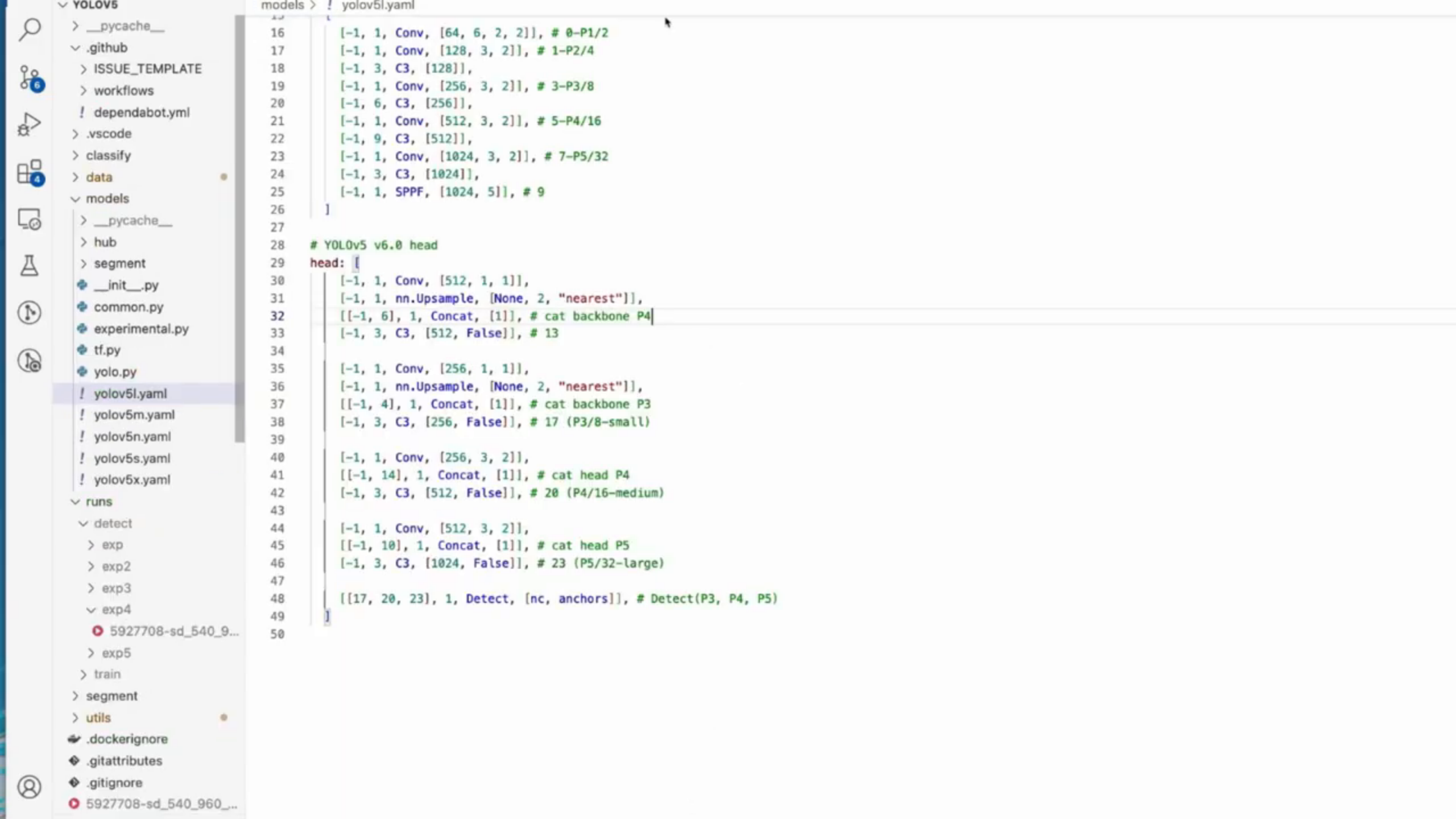

1×64×160×160
Conv
W (64×64×1×1)
B (64)
Sigmoid
Mul
W (64×64×3×3)
W (64×128×1×1)
Add
Concat
W (128×128×1×1)
B (128)
W (256×128×3×3)
B (256)
1×256×80×80

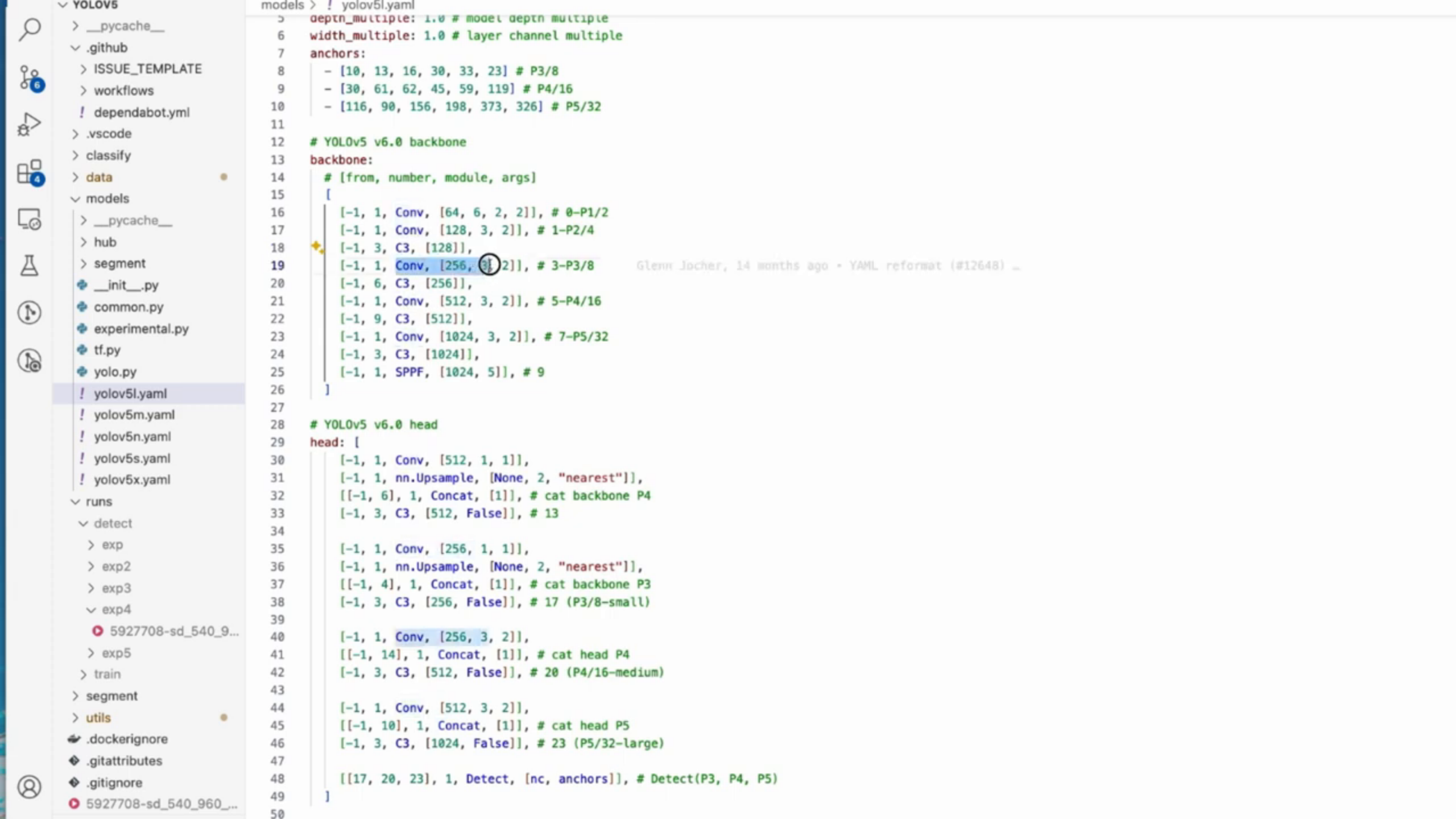

YoloV5_1.key — Edited
View
Zoom
Add Slide
Play
Table
Chart
Text
Shape
Media
Comment
Share
Format
Animate
Document
Backbone
Neck
1x256x80x80
1x512x40x40
1x1024x20x20
Conv
C3
SPPF
Concat
Max Pool
BN
SiLU
SiLU = x * sigmoid(x)

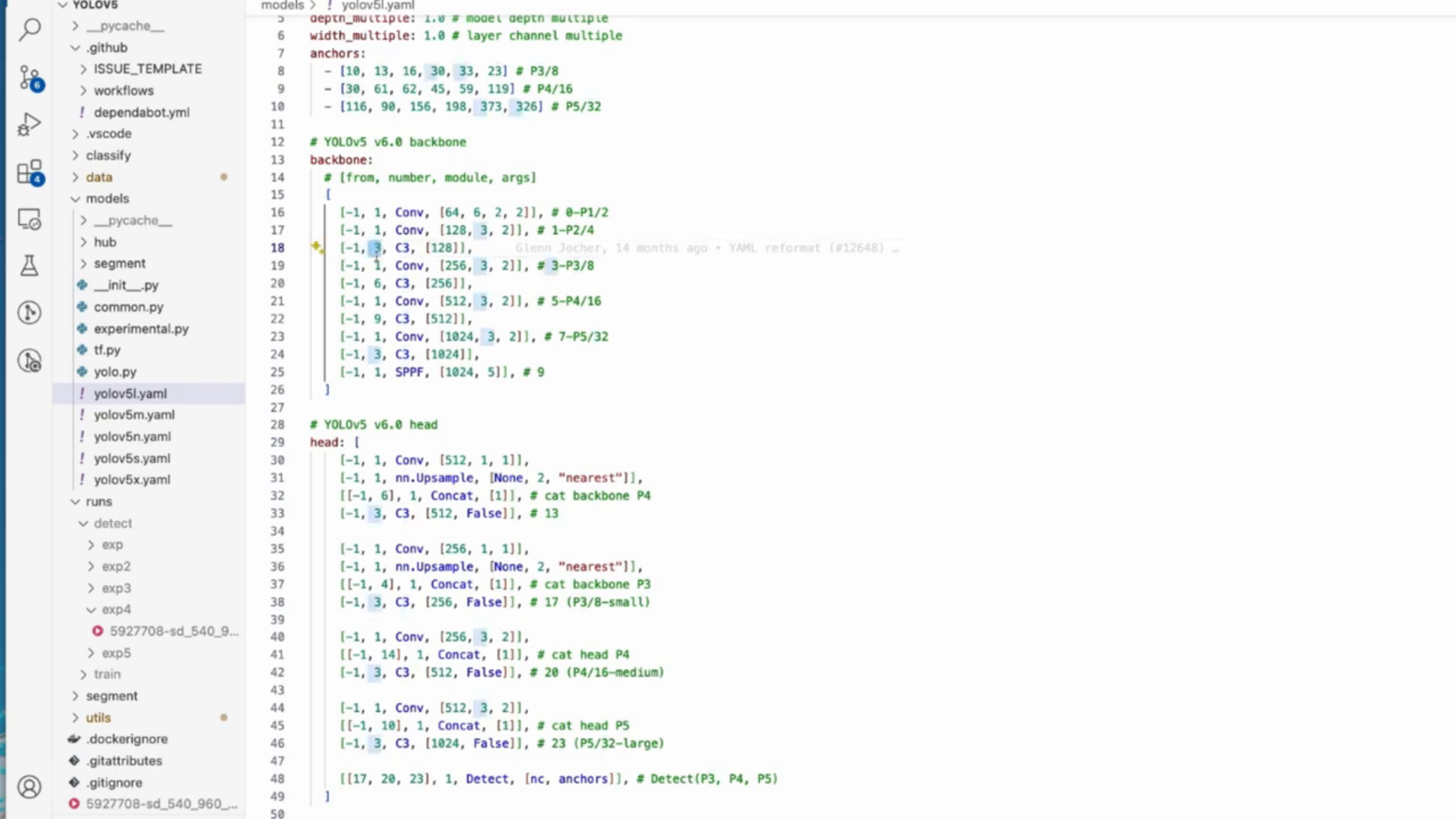

1×128×160×160
Sigmoid
Mul
Conv
W (256×128×3×3)
B (256)
1×256×80×80
1×128×80×80
W (128×256×1×1)
B (128)
W (128×128×1×1)
W (128×128×3×3)

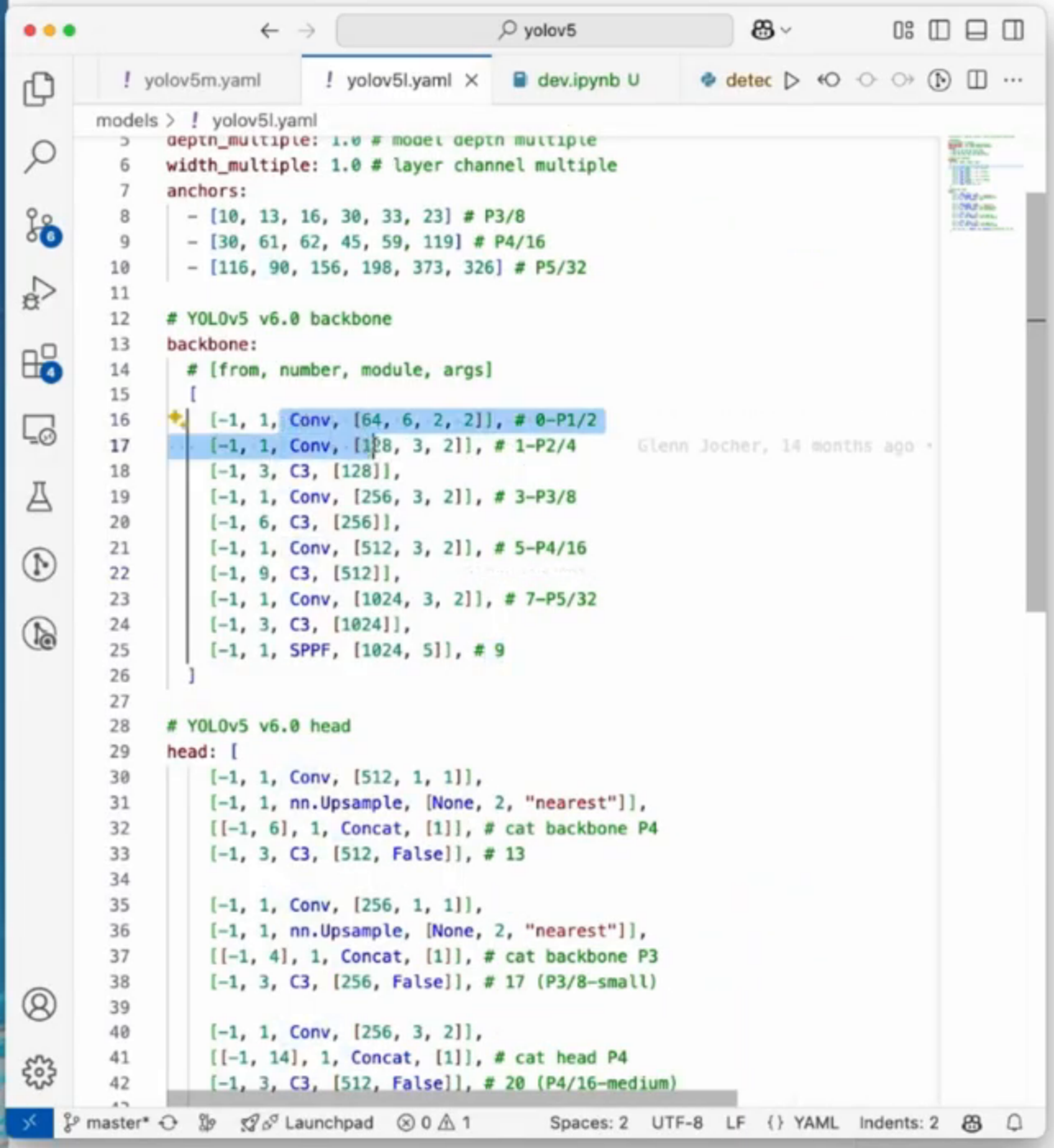

YoloV5_1.key — Edited
Backbone
Neck
1x256x80x80
1x512x40x40
1x1024x20x20
Conv
C3
SPPF
N
Concat
Max Pool
Conv → BN → SiLU
SiLU = x * sigmoid(x)

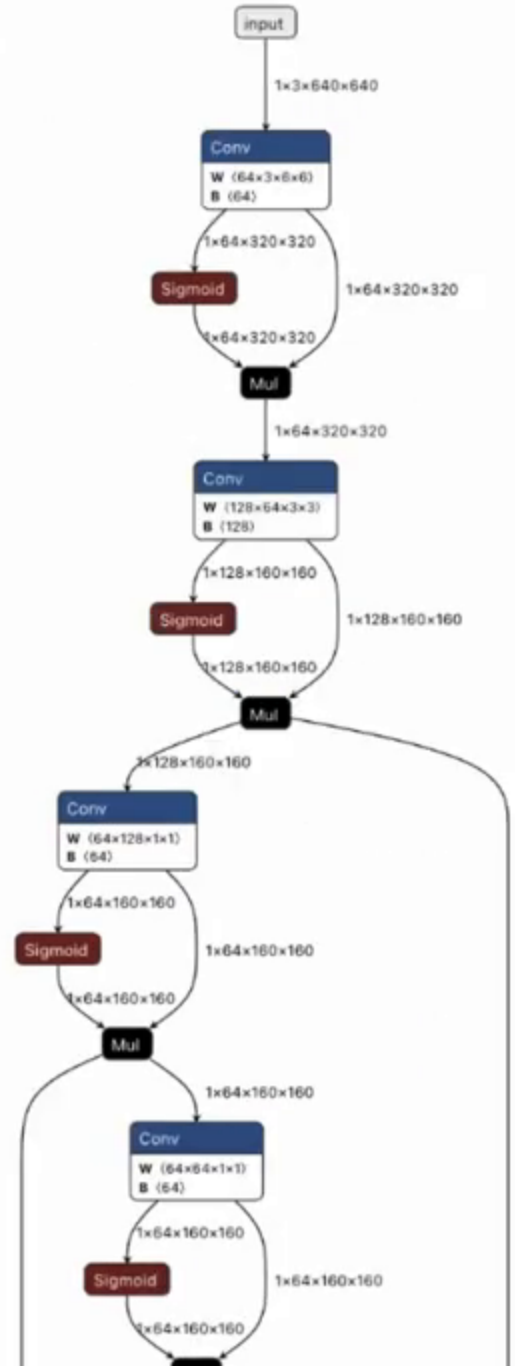

W (128×128×3×3)
B (128)
1×128×80×80
Sigmoid
Mul
Add
W (128×128×1×1)
W (128×256×1×1)
W (256×256×1×1)
B (256)
Concat
1×256×80×80

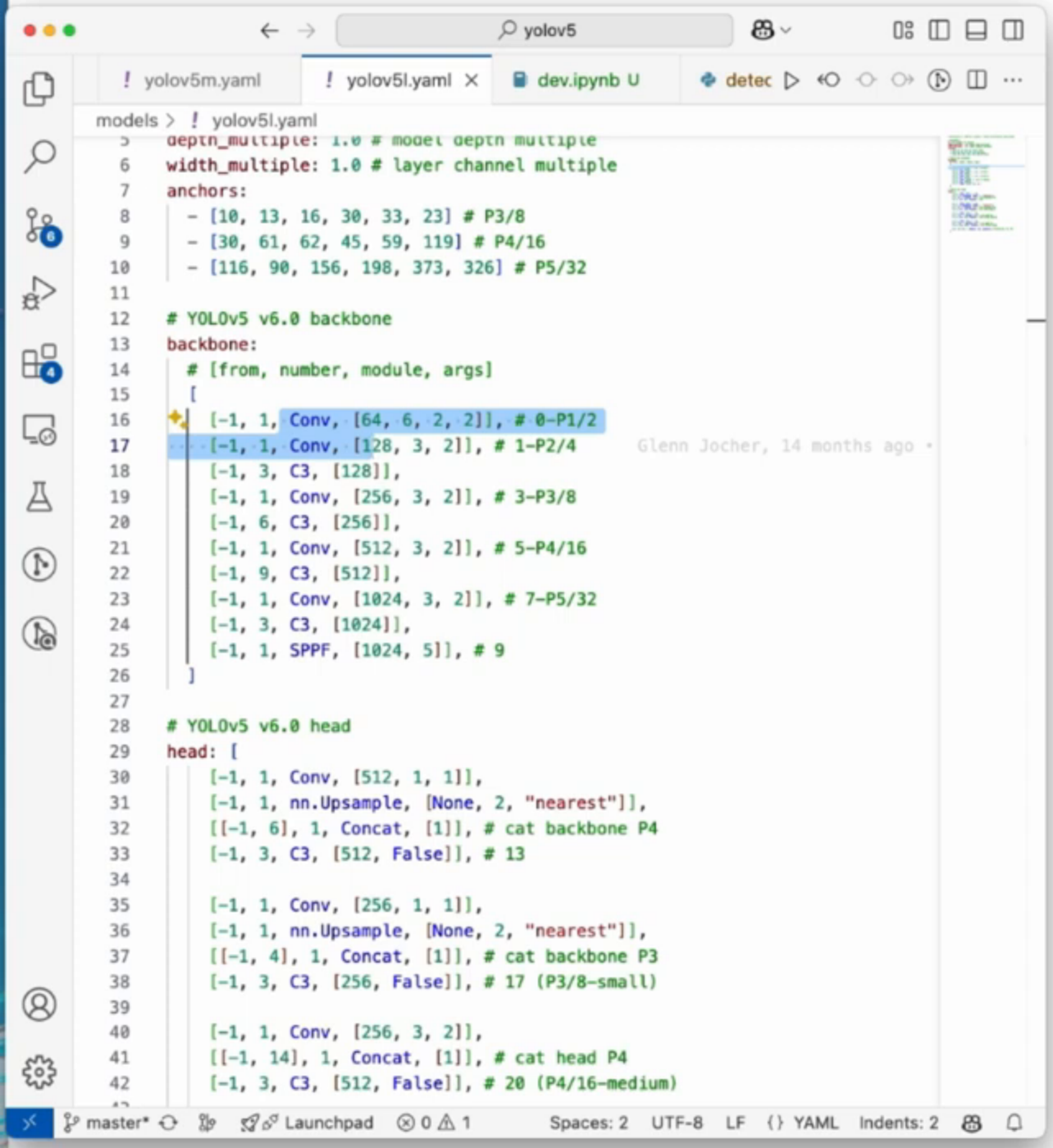

YOLOV5
__pycache__
.github
ISSUE_TEMPLATE
workflows
dependabot.yml
.vscode
classify
data
models
__pycache__
hub
segment
__init__.py
common.py
experimental.py
tf.py
yolo.py
yolov5l.yaml
yolov5m.yaml
yolov5n.yaml
yolov5s.yaml
yolov5x.yaml
runs
detect
exp
exp2
exp3
exp4
5927708-sd_540_9...
exp5
train
segment
utils
.dockerignore
.gitattributes
.gitignore
5927708-sd_540_960_...
models > ! yolov5l.yaml
[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
[-1, 1, Conv, [128, 3, 2]], # 1-P2/4
[-1, 3, C3, [128]],
[-1, 1, Conv, [256, 3, 2]], # 3-P3/8
[-1, 6, C3, [256]],
[-1, 1, Conv, [512, 3, 2]], # 5-P4/16
[-1, 9, C3, [512]],
[-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
[-1, 3, C3, [1024]],
[-1, 1, SPPF, [1024, 5]], # 9
]
YOLOv5 v6.0 head
head: [
[-1, 1, Conv, [512, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 6], 1, Concat, [1]], # cat backbone P4
[-1, 3, C3, [512, False]], # 13
[-1, 1, Conv, [256, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 4], 1, Concat, [1]], # cat backbone P3
[-1, 3, C3, [256, False]], # 17 (P3/8-small)
[-1, 1, Conv, [256, 3, 2]],
[[-1, 14], 1, Concat, [1]], # cat head P4
[-1, 3, C3, [512, False]], # 20 (P4/16-medium)
[-1, 1, Conv, [512, 3, 2]],
[[-1, 10], 1, Concat, [1]], # cat head P5
[-1, 3, C3, [1024, False]], # 23 (P5/32-large)
[[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
]

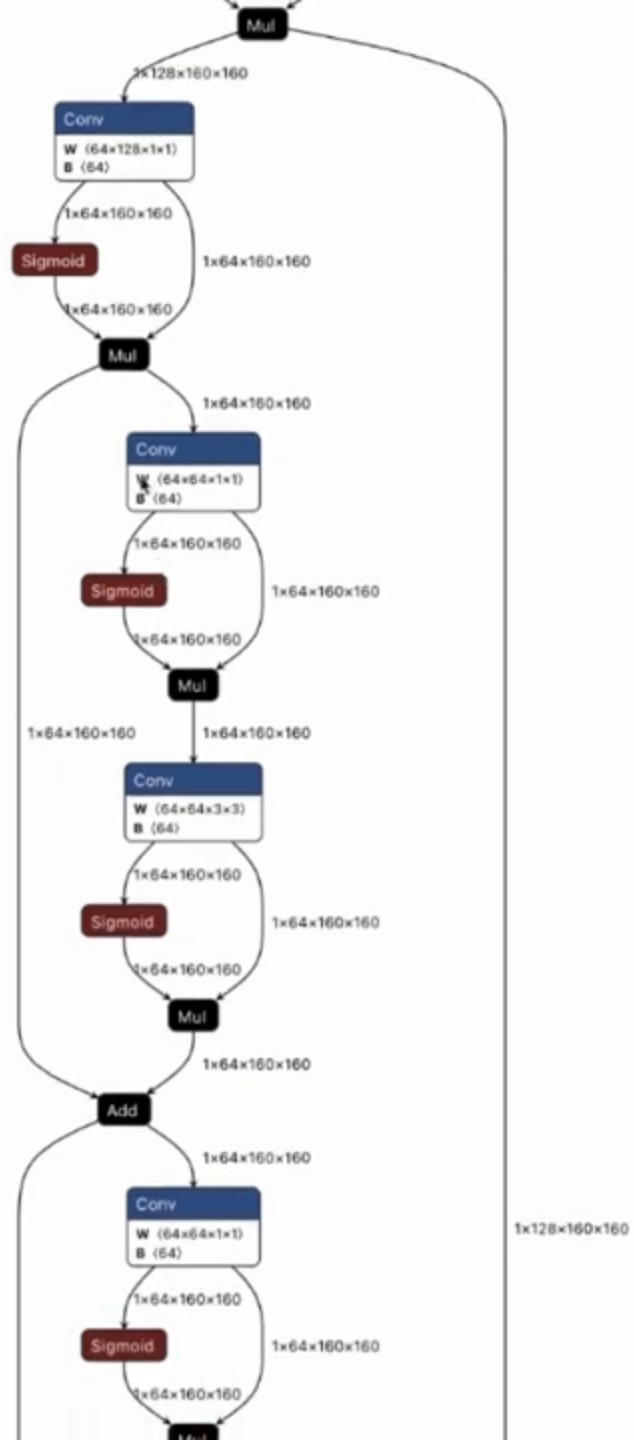

YOLOV5
__pycache__
.github
ISSUE_TEMPLATE
workflows
dependabot.yml
.vscode
classify
data
models
__pycache__
hub
segment
__init__.py
common.py
experimental.py
tf.py
yolo.py
yolov5l.yaml
yolov5m.yaml
yolov5n.yaml
yolov5s.yaml
yolov5x.yaml
runs
detect
exp
exp2
exp3
exp4
5927708-sd_540_9...
exp5
train
segment
utils
.dockerignore
.gitattributes
.gitignore
5927708-sd_540_960_...
models > ! yolov5l.yaml
depth_multiple: 1.0 # model depth multiple
width_multiple: 1.0 # layer channel multiple
anchors:
- [10, 13, 16, 30, 33, 23] # P3/8
- [30, 61, 62, 45, 59, 119] # P4/16
- [116, 90, 156, 198, 373, 326] # P5/32
YOLOv5 v6.0 backbone
backbone:
[from, number, module, args]
[
[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
[-1, 1, Conv, [128, 3, 2]], # 1-P2/4
[-1, 3, C3, [128]],
[-1, 1, Conv, [256, 3, 2]], # 3-P3/8
Glenn Jocher, 14 months ago • YAML reformat (#12648) …
[-1, 6, C3, [256]],
[-1, 1, Conv, [512, 3, 2]], # 5-P4/16
[-1, 9, C3, [512]],
[-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
[-1, 3, C3, [1024]],
[-1, 1, SPPF, [1024, 5]], # 9
]
YOLOv5 v6.0 head
head: [
[-1, 1, Conv, [512, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 6], 1, Concat, [1]], # cat backbone P4
[-1, 3, C3, [512, False]], # 13
[-1, 1, Conv, [256, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 4], 1, Concat, [1]], # cat backbone P3
[-1, 3, C3, [256, False]], # 17 (P3/8-small)
[-1, 1, Conv, [256, 3, 2]],
[[-1, 14], 1, Concat, [1]], # cat head P4
[-1, 3, C3, [512, False]], # 20 (P4/16-medium)
[-1, 1, Conv, [512, 3, 2]],
[[-1, 10], 1, Concat, [1]], # cat head P5
[-1, 3, C3, [1024, False]], # 23 (P5/32-large)
[[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
]

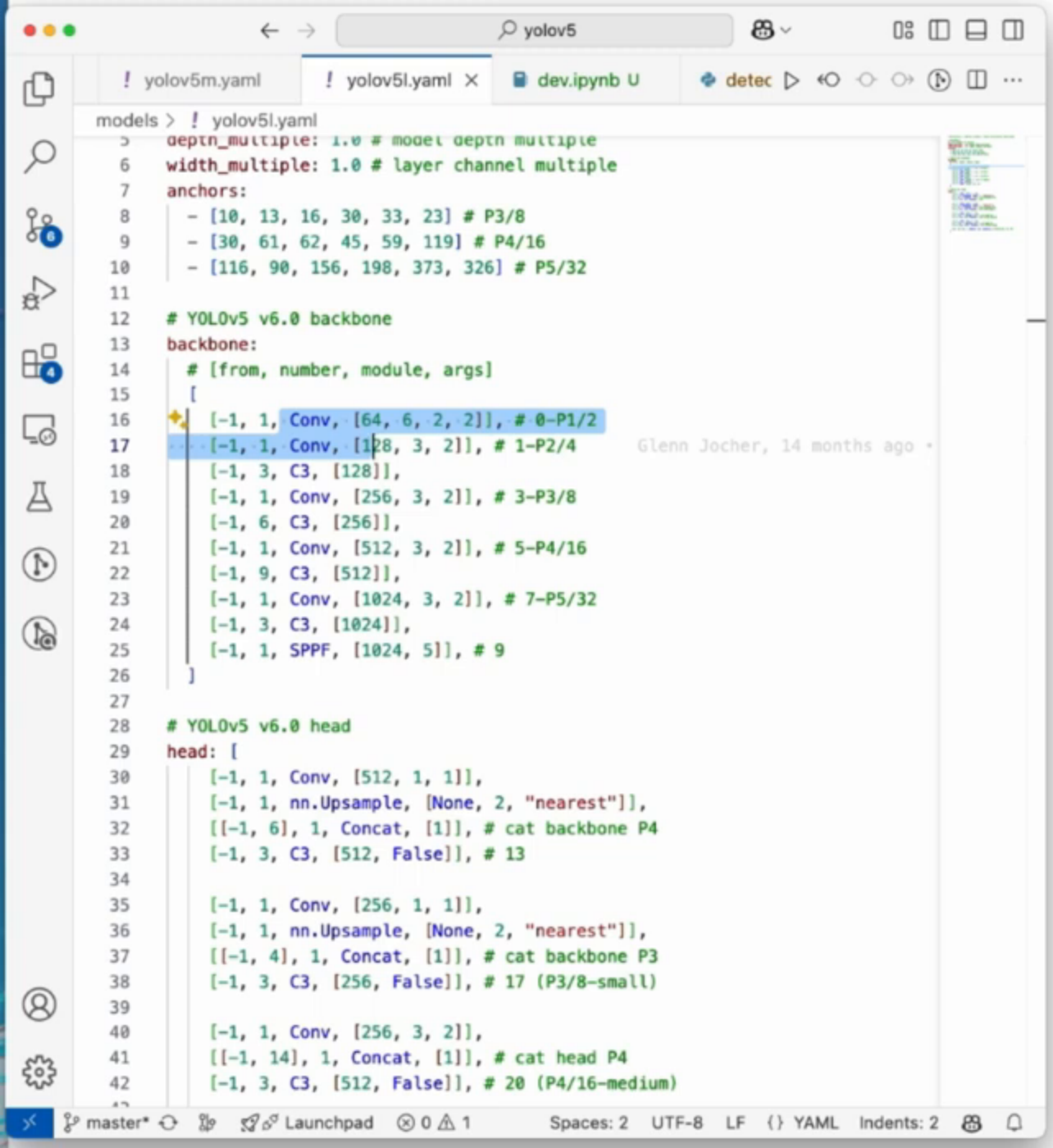

YOLOV5
__pycache__
.github
ISSUE_TEMPLATE
workflows
dependabot.yml
.vscode
classify
data
models
__pycache__
hub
segment
__init__.py
common.py
experimental.py
tf.py
yolo.py
yolov5l.yaml
yolov5m.yaml
yolov5n.yaml
yolov5s.yaml
yolov5x.yaml
runs
detect
exp
exp2
exp3
exp4
5927708-sd_540_9...
exp5
train
segment
utils
.dockerignore
.gitattributes
.gitignore
5927708-sd_540_960_...
models > ! yolov5l.yaml
depth_multiple: 1.0 # model depth multiple
width_multiple: 1.0 # layer channel multiple
anchors:
- [10, 13, 16, 30, 33, 23] # P3/8
- [30, 61, 62, 45, 59, 119] # P4/16
- [116, 90, 156, 198, 373, 326] # P5/32
YOLOv5 v6.0 backbone
backbone:
[from, number, module, args]
[
[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
[-1, 1, Conv, [128, 3, 2]], # 1-P2/4
[-1, 3, C3, [128]],
Glenn Jocher, 14 months ago • YAML reformat (#12648) …
[-1, 1, Conv, [256, 3, 2]], # 3-P3/8
[-1, 6, C3, [256]],
[-1, 1, Conv, [512, 3, 2]], # 5-P4/16
[-1, 9, C3, [512]],
[-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
[-1, 3, C3, [1024]],
[-1, 1, SPPF, [1024, 5]], # 9
]
YOLOv5 v6.0 head
head: [
[-1, 1, Conv, [512, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 6], 1, Concat, [1]], # cat backbone P4
[-1, 3, C3, [512, False]], # 13
[-1, 1, Conv, [256, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 4], 1, Concat, [1]], # cat backbone P3
[-1, 3, C3, [256, False]], # 17 (P3/8-small)
[-1, 1, Conv, [256, 3, 2]],
[[-1, 14], 1, Concat, [1]], # cat head P4
[-1, 3, C3, [512, False]], # 20 (P4/16-medium)
[-1, 1, Conv, [512, 3, 2]],
[[-1, 10], 1, Concat, [1]], # cat head P5
[-1, 3, C3, [1024, False]], # 23 (P5/32-large)
[[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
]

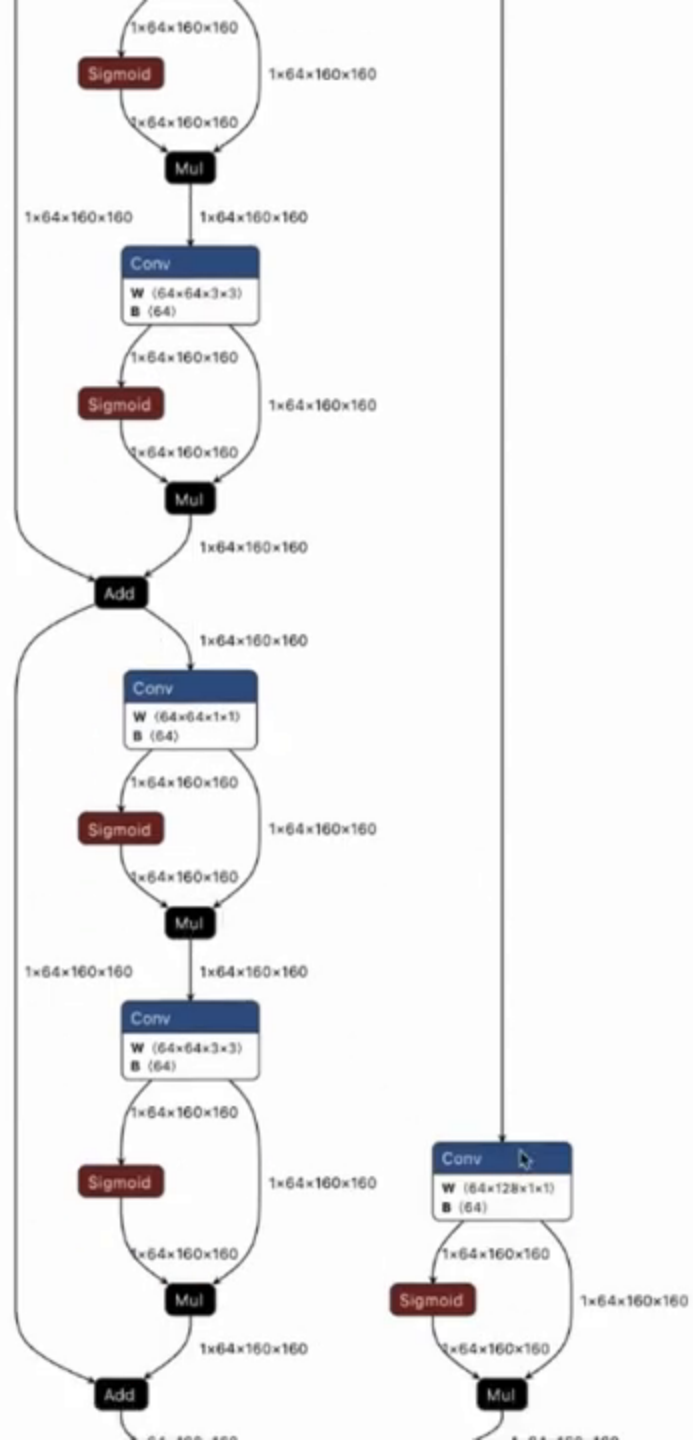

yolov5
yolov5m.yaml
yolov5l.yaml
dev.ipynb U
models > yolov5l.yaml
depth_multiple: 1.0 # model depth multiple
width_multiple: 1.0 # layer channel multiple
anchors:
- [10, 13, 16, 30, 33, 23] # P3/8
- [30, 61, 62, 45, 59, 119] # P4/16
- [116, 90, 156, 198, 373, 326] # P5/32
YOLOv5 v6.0 backbone
backbone:
[from, number, module, args]
[
[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
[-1, 1, Conv, [128, 3, 2]], # 1-P2/4
Glenn Jocher, 14 months ago
[-1, 3, C3, [128]],
[-1, 1, Conv, [256, 3, 2]], # 3-P3/8
[-1, 6, C3, [256]],
[-1, 1, Conv, [512, 3, 2]], # 5-P4/16
[-1, 9, C3, [512]],
[-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
[-1, 3, C3, [1024]],
[-1, 1, SPPF, [1024, 5]], # 9
]
YOLOv5 v6.0 head
head: [
[-1, 1, Conv, [512, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 6], 1, Concat, [1]], # cat backbone P4
[-1, 3, C3, [512, False]], # 13
[-1, 1, Conv, [256, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 4], 1, Concat, [1]], # cat backbone P3
[-1, 3, C3, [256, False]], # 17 (P3/8-small)
[-1, 1, Conv, [256, 3, 2]],
[[-1, 14], 1, Concat, [1]], # cat head P4
[-1, 3, C3, [512, False]], # 20 (P4/16-medium)
master*
Launchpad
Spaces: 2
UTF-8
LF
YAML
Indents: 2

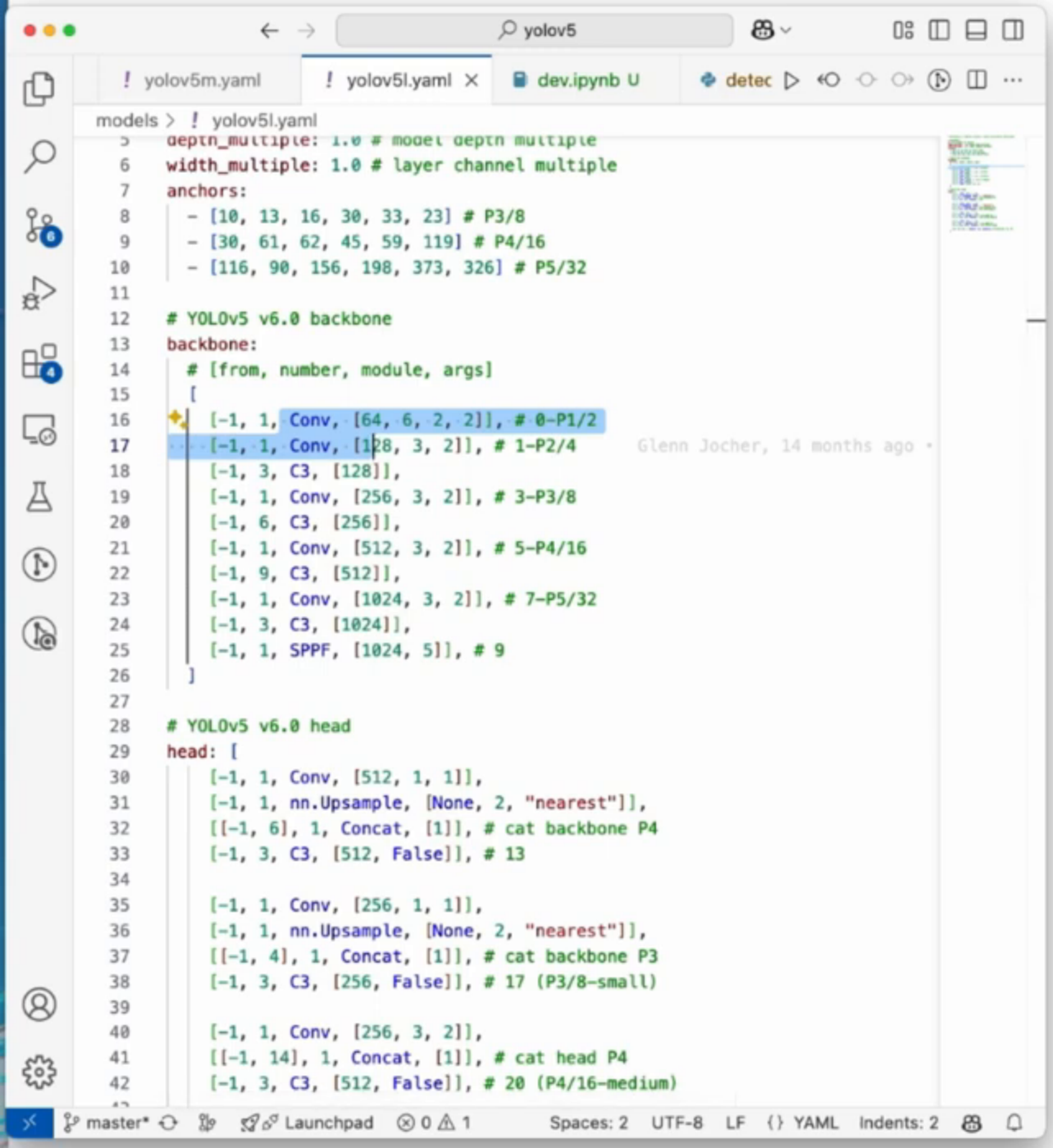

input
1×3×640×640
Conv
W (64×3×6×6)
B (64)
1×64×320×320
Sigmoid
1×64×320×320
1×64×320×320
Mul
1×64×320×320
Conv
W (128×64×3×3)
B (128)
1×128×160×160
Sigmoid
1×128×160×160
1×128×160×160
Mul
1×128×160×160
Conv
W (64×128×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×64×160×160
Conv
W (64×64×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160

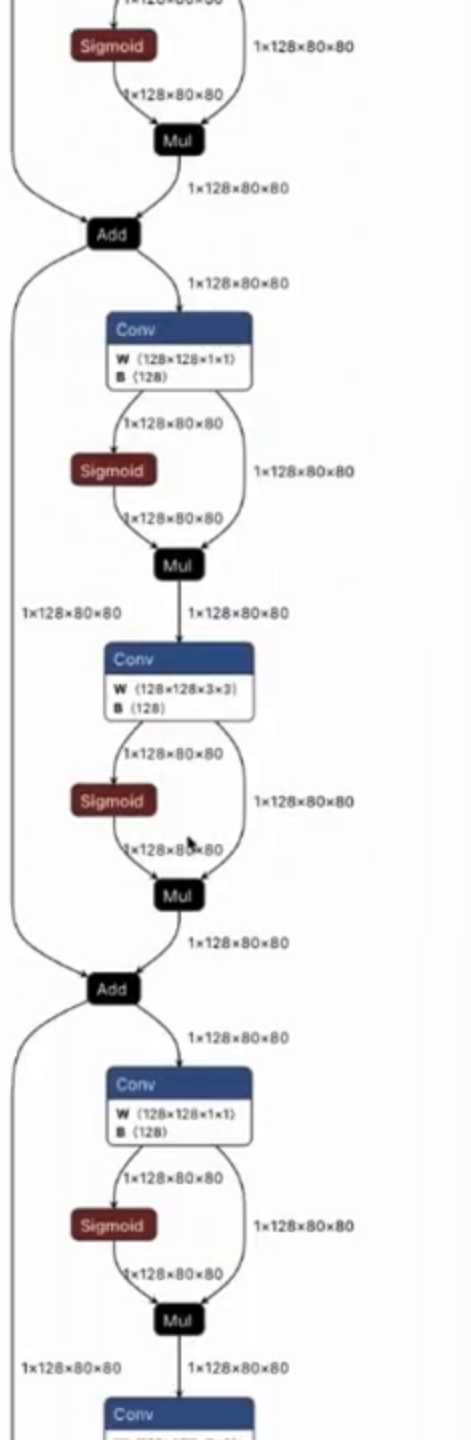

yolov5
yolov5m.yaml
yolov5l.yaml
dev.ipynb U
detec
models > yolov5l.yaml
depth_multiple: 1.0 # model depth multiple
width_multiple: 1.0 # layer channel multiple
anchors:
- [10, 13, 16, 30, 33, 23] # P3/8
- [30, 61, 62, 45, 59, 119] # P4/16
- [116, 90, 156, 198, 373, 326] # P5/32
YOLOv5 v6.0 backbone
backbone:
[from, number, module, args]
[
[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
[-1, 1, Conv, [128, 3, 2]], # 1-P2/4
Glenn Jocher, 14 months ago
[-1, 3, C3, [128]],
[-1, 1, Conv, [256, 3, 2]], # 3-P3/8
[-1, 6, C3, [256]],
[-1, 1, Conv, [512, 3, 2]], # 5-P4/16
[-1, 9, C3, [512]],
[-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
[-1, 3, C3, [1024]],
[-1, 1, SPPF, [1024, 5]], # 9
]
YOLOv5 v6.0 head
head: [
[-1, 1, Conv, [512, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 6], 1, Concat, [1]], # cat backbone P4
[-1, 3, C3, [512, False]], # 13
[-1, 1, Conv, [256, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 4], 1, Concat, [1]], # cat backbone P3
[-1, 3, C3, [256, False]], # 17 (P3/8-small)
[-1, 1, Conv, [256, 3, 2]],
[[-1, 14], 1, Concat, [1]], # cat head P4
[-1, 3, C3, [512, False]], # 20 (P4/16-medium)
master*
Launchpad
0 1
Spaces: 2
UTF-8
LF
YAML
Indents: 2

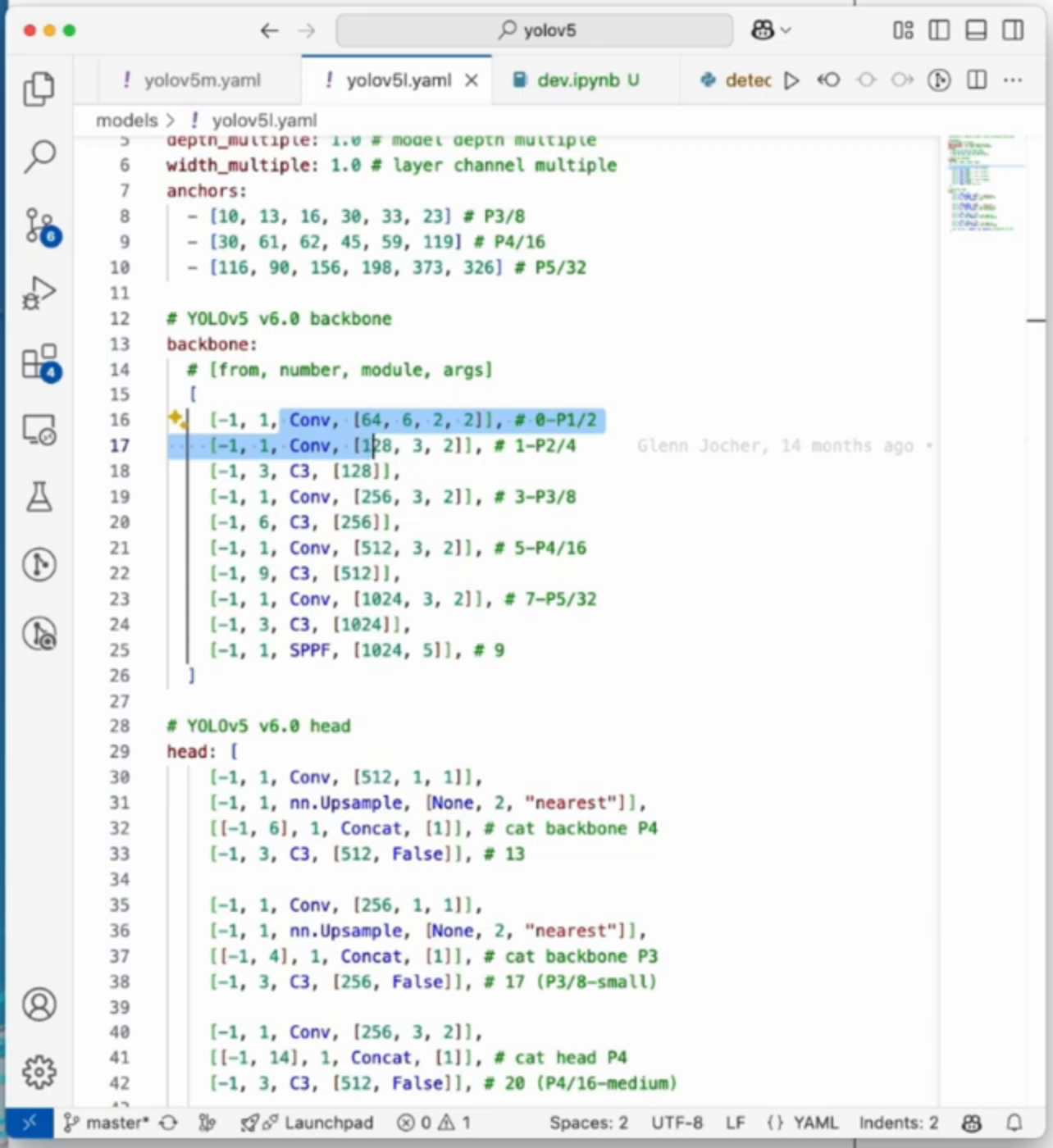

Mul
1×128×160×160
Conv
W (64×128×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×64×160×160
Conv
W (64×64×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×64×160×160
1×64×160×160
Conv
W (64×64×3×3)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×64×160×160
Add
1×64×160×160
Conv
W (64×64×1×1)
B (64)
1×64×160×160
Sigmoid
1×64×160×160
1×64×160×160
Mul
1×128×160×160

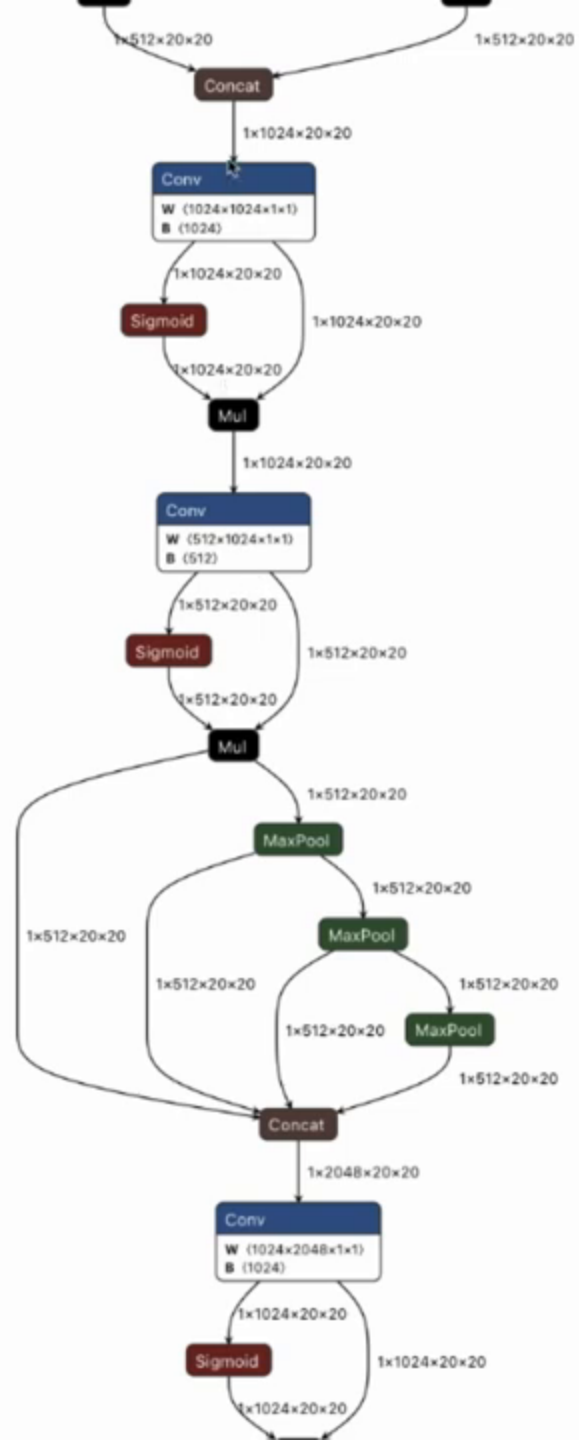

yolov5
yolov5m.yaml
yolov5l.yaml
dev.ipynb U
detec
models > yolov5l.yaml
depth_multiple: 1.0 # model depth multiple
width_multiple: 1.0 # layer channel multiple
anchors:
- [10, 13, 16, 30, 33, 23] # P3/8
- [30, 61, 62, 45, 59, 119] # P4/16
- [116, 90, 156, 198, 373, 326] # P5/32
YOLOv5 v6.0 backbone
backbone:
[from, number, module, args]
[
[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
[-1, 1, Conv, [128, 3, 2]], # 1-P2/4
Glenn Jocher, 14 months ago
[-1, 3, C3, [128]],
[-1, 1, Conv, [256, 3, 2]], # 3-P3/8
[-1, 6, C3, [256]],
[-1, 1, Conv, [512, 3, 2]], # 5-P4/16
[-1, 9, C3, [512]],
[-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
[-1, 3, C3, [1024]],
[-1, 1, SPPF, [1024, 5]], # 9
]
YOLOv5 v6.0 head
head: [
[-1, 1, Conv, [512, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 6], 1, Concat, [1]], # cat backbone P4
[-1, 3, C3, [512, False]], # 13
[-1, 1, Conv, [256, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 4], 1, Concat, [1]], # cat backbone P3
[-1, 3, C3, [256, False]], # 17 (P3/8-small)
[-1, 1, Conv, [256, 3, 2]],
[[-1, 14], 1, Concat, [1]], # cat head P4
[-1, 3, C3, [512, False]], # 20 (P4/16-medium)
master*
Launchpad
0 1
Spaces: 2
UTF-8
LF
YAML
Indents: 2

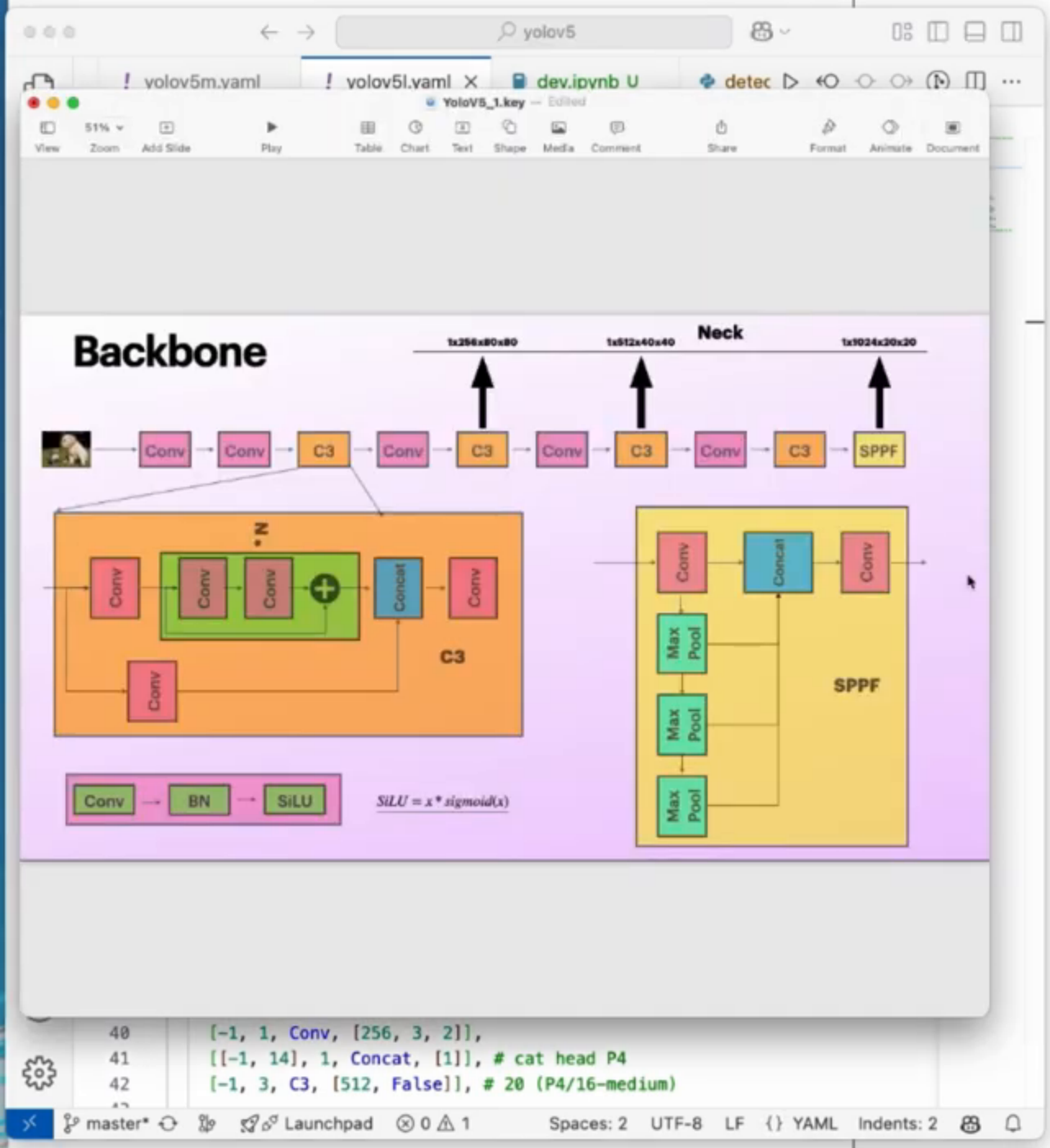

1×64×160×160
Sigmoid
Mul
Conv
W (64×64×3×3)
B (64)
Add
W (64×64×1×1)
W (64×128×1×1)


yolov5
yolov5m.yaml
yolov5l.yaml
dev.ipynb U
detec
models > yolov5l.yaml
depth_multiple: 1.0 # model depth multiple
width_multiple: 1.0 # layer channel multiple
anchors:
- [10, 13, 16, 30, 33, 23] # P3/8
- [30, 61, 62, 45, 59, 119] # P4/16
- [116, 90, 156, 198, 373, 326] # P5/32
YOLOv5 v6.0 backbone
backbone:
[from, number, module, args]
[
[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
[-1, 1, Conv, [128, 3, 2]], # 1-P2/4
Glenn Jocher, 14 months ago
[-1, 3, C3, [128]],
[-1, 1, Conv, [256, 3, 2]], # 3-P3/8
[-1, 6, C3, [256]],
[-1, 1, Conv, [512, 3, 2]], # 5-P4/16
[-1, 9, C3, [512]],
[-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
[-1, 3, C3, [1024]],
[-1, 1, SPPF, [1024, 5]], # 9
]
YOLOv5 v6.0 head
head: [
[-1, 1, Conv, [512, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 6], 1, Concat, [1]], # cat backbone P4
[-1, 3, C3, [512, False]], # 13
[-1, 1, Conv, [256, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 4], 1, Concat, [1]], # cat backbone P3
[-1, 3, C3, [256, False]], # 17 (P3/8-small)
[-1, 1, Conv, [256, 3, 2]],
[[-1, 14], 1, Concat, [1]], # cat head P4
[-1, 3, C3, [512, False]], # 20 (P4/16-medium)
master*
Launchpad
0 1
Spaces: 2
UTF-8
LF
YAML
Indents: 2

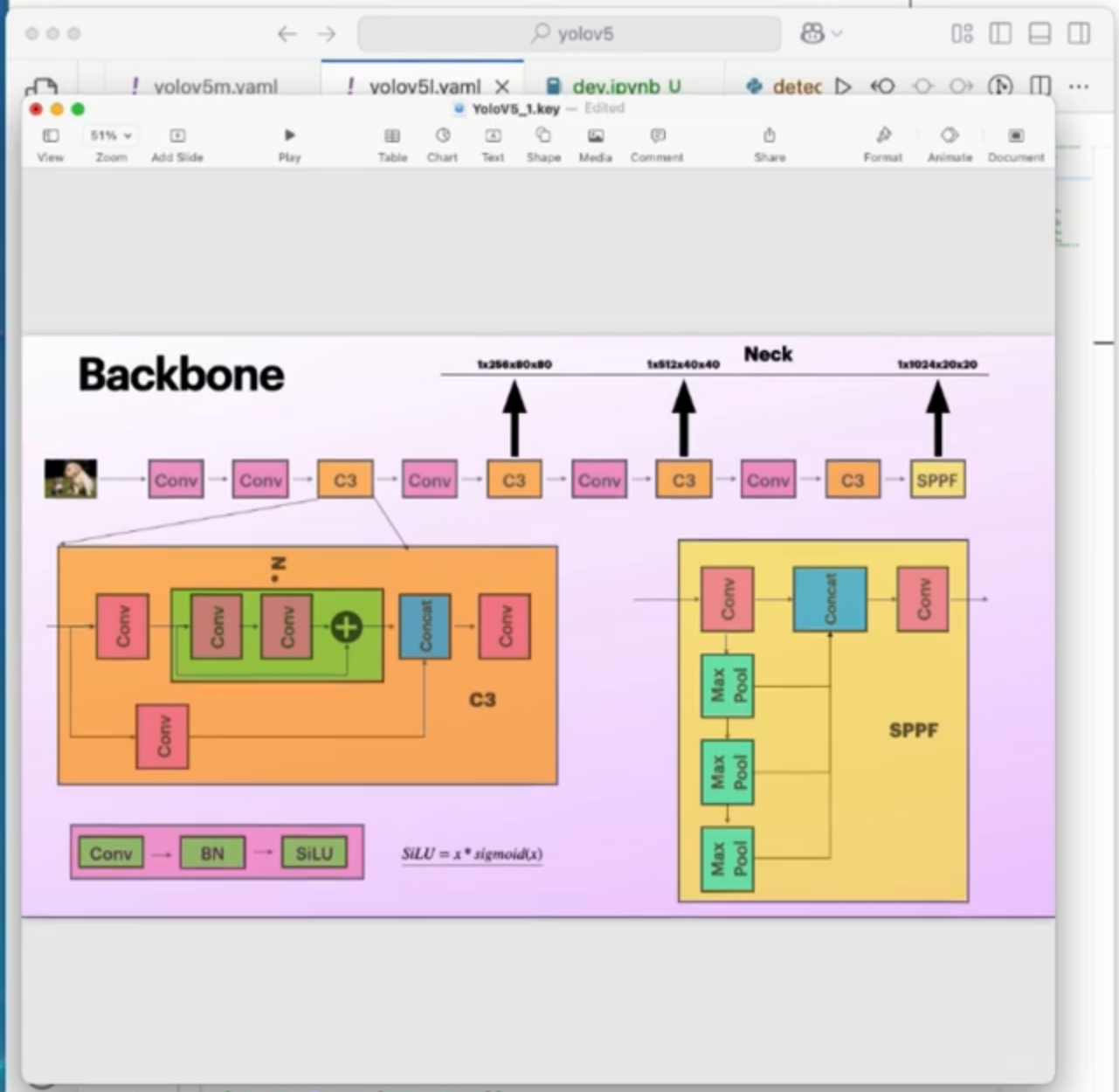

Sigmoid
1×128×80×80
Mul
Add
Conv
W (128×128×1×1)
B (128)
W (128×128×3×3)


yolov5
yolov5m.yaml
yolov5l.yaml
dev.ipynb U
detec
models > yolov5l.yaml
depth_multiple: 1.0 # model depth multiple
width_multiple: 1.0 # layer channel multiple
anchors:
- [10, 13, 16, 30, 33, 23] # P3/8
- [30, 61, 62, 45, 59, 119] # P4/16
- [116, 90, 156, 198, 373, 326] # P5/32
YOLOv5 v6.0 backbone
backbone:
[from, number, module, args]
[
[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
[-1, 1, Conv, [128, 3, 2]], # 1-P2/4
Glenn Jocher, 14 months ago
[-1, 3, C3, [128]],
[-1, 1, Conv, [256, 3, 2]], # 3-P3/8
[-1, 6, C3, [256]],
[-1, 1, Conv, [512, 3, 2]], # 5-P4/16
[-1, 9, C3, [512]],
[-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
[-1, 3, C3, [1024]],
[-1, 1, SPPF, [1024, 5]], # 9
]
YOLOv5 v6.0 head
head: [
[-1, 1, Conv, [512, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 6], 1, Concat, [1]], # cat backbone P4
[-1, 3, C3, [512, False]], # 13
[-1, 1, Conv, [256, 1, 1]],
[-1, 1, nn.Upsample, [None, 2, "nearest"]],
[[-1, 4], 1, Concat, [1]], # cat backbone P3
[-1, 3, C3, [256, False]], # 17 (P3/8-small)
[-1, 1, Conv, [256, 3, 2]],
[[-1, 14], 1, Concat, [1]], # cat head P4
[-1, 3, C3, [512, False]], # 20 (P4/16-medium)
master*
Launchpad
0 1
Spaces: 2
UTF-8
LF
YAML
Indents: 2

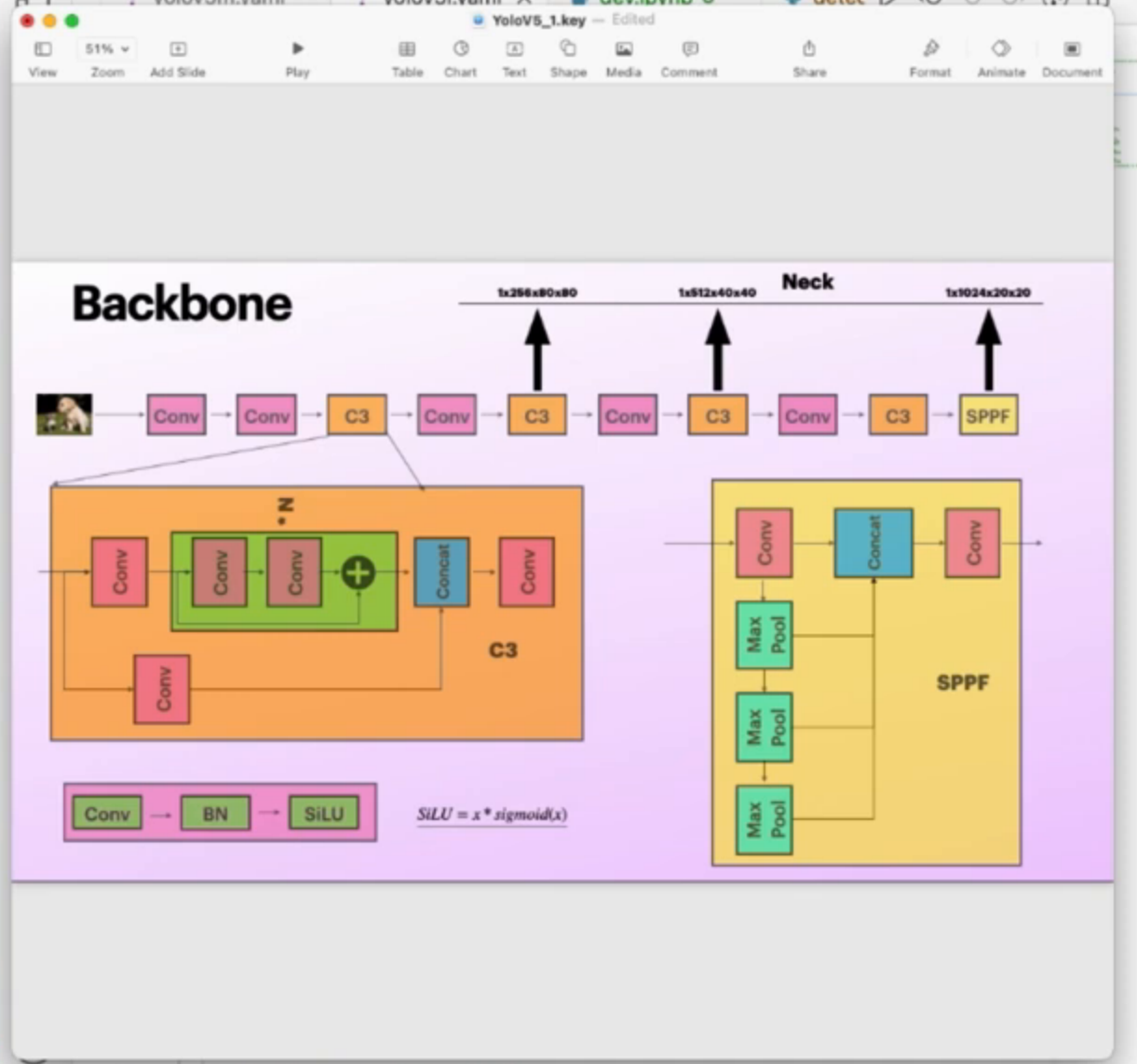

1×512×20×20
1×512×20×20
Concat
1×1024×20×20
Conv
W (1024×1024×1×1)
B (1024)
1×1024×20×20
Sigmoid
1×1024×20×20
1×1024×20×20
Mul
1×1024×20×20
Conv
W (512×1024×1×1)
B (512)
1×512×20×20
Sigmoid
1×512×20×20
1×512×20×20
Mul
1×512×20×20
MaxPool
1×512×20×20
1×512×20×20
MaxPool
1×512×20×20
1×512×20×20
1×512×20×20
MaxPool
1×512×20×20
Concat
1×2048×20×20
Conv
W (1024×2048×1×1)
B (1024)
1×1024×20×20
Sigmoid
1×1024×20×20
1×1024×20×20

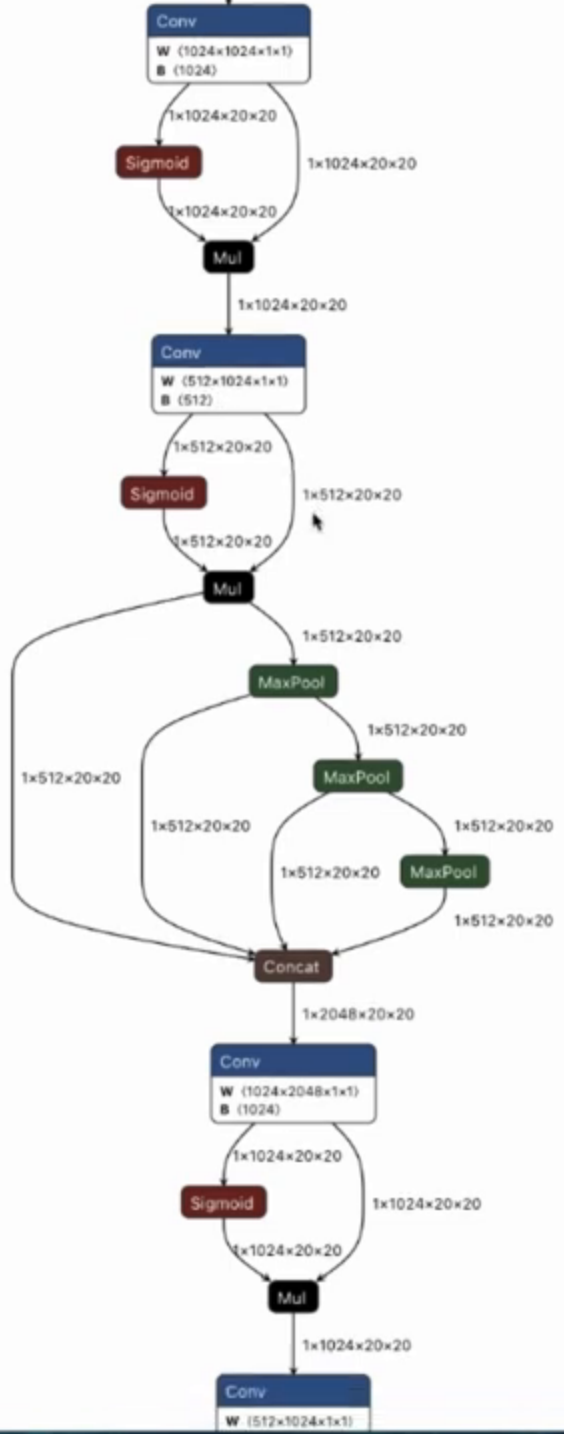

yolov5
yolov5m.yaml
yolov5l.yaml
dev.ipynb U
YoloV5_1.key — Edited
Backbone
Neck
1x256x80x80
1x512x40x40
1x1024x20x20
Conv
C3
SPPF
Concat
Max Pool
BN
SiLU
SiLU = x * sigmoid(x)
40 [-1, 1, Conv, [256, 3, 2]],
41 [[-1, 14], 1, Concat, [1]], # cat head P4
42 [-1, 3, C3, [512, False]], # 20 (P4/16-medium)
master*
Launchpad
Spaces: 2
UTF-8
LF
YAML
Indents: 2


1×512×20×20
Concat
1×1024×20×20
Conv
W (1024×1024×1×1)
B (1024)
Sigmoid
Mul
W (512×1024×1×1)
B (512)
MaxPool
W (1024×2048×1×1)
B (1024)


yolov5
yolov5m.yaml
yolov5l.yaml
dev.ipynb U
YoloV5_1.key — Edited
Backbone
Neck
1x256x80x80
1x512x40x40
1x1024x20x20
Conv
C3
SPPF
N *
C3
SPPF
Max Pool
Concat
Conv → BN → SiLU
SiLU = x * sigmoid(x)
40 [-1, 1, Conv, [256, 3, 2]],
41 [[-1, 14], 1, Concat, [1]], # cat head P4
42 [-1, 3, C3, [512, False]], # 20 (P4/16-medium)
master*
Launchpad
Spaces: 2
UTF-8
LF
YAML
Indents: 2

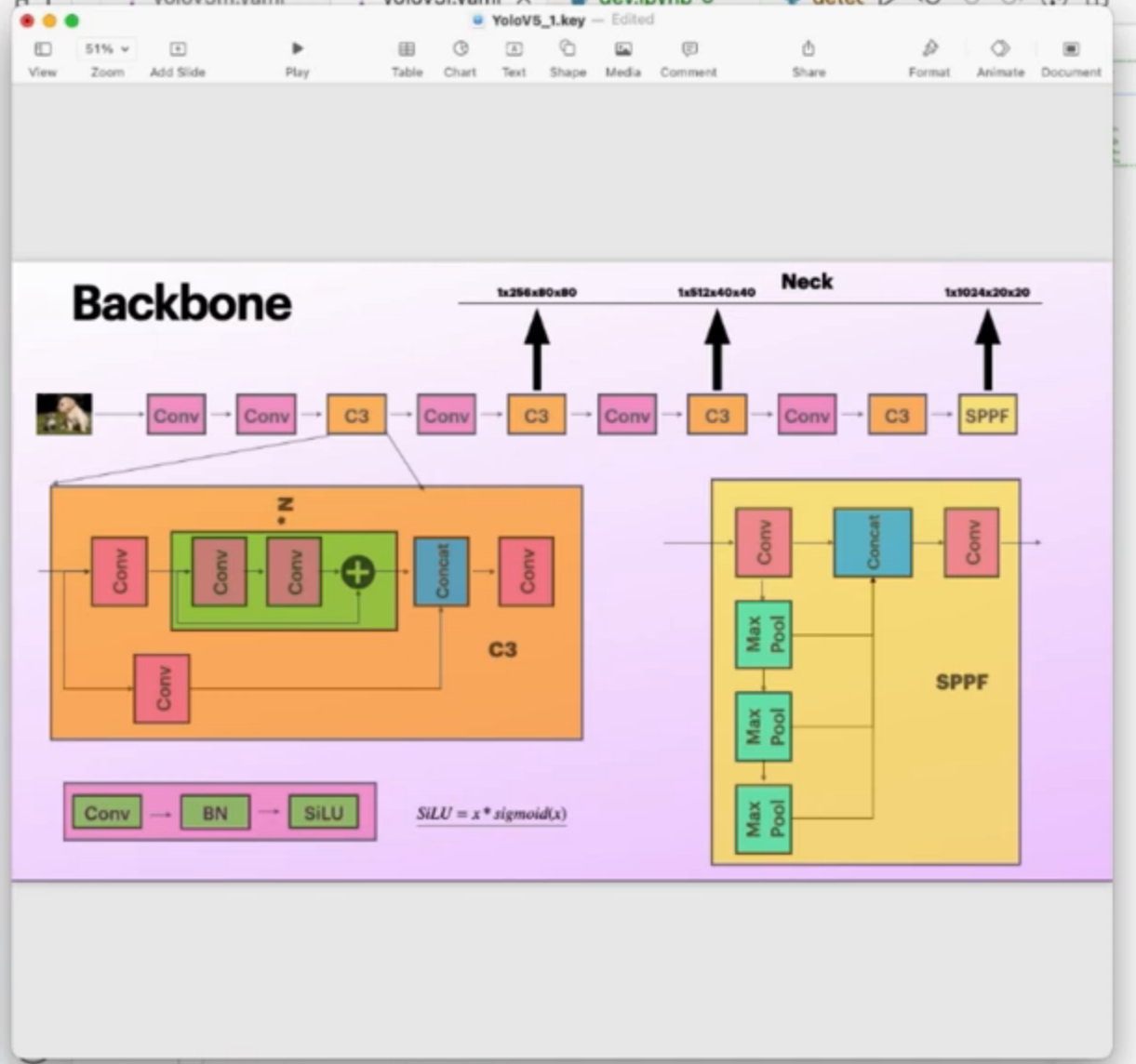

1×512×20×20
Concat
1×1024×20×20
Conv
W (1024×1024×1×1)
B (1024)
Sigmoid
Mul
W (512×1024×1×1)
B (512)
MaxPool
W (1024×2048×1×1)






```
40    [-1, 1, Conv, [256, 3, 2]],
41    [[-1, 14], 1, Concat, [1]], # cat head P4
42    [-1, 3, C3, [512, False]], # 20 (P4/16-medium)
```

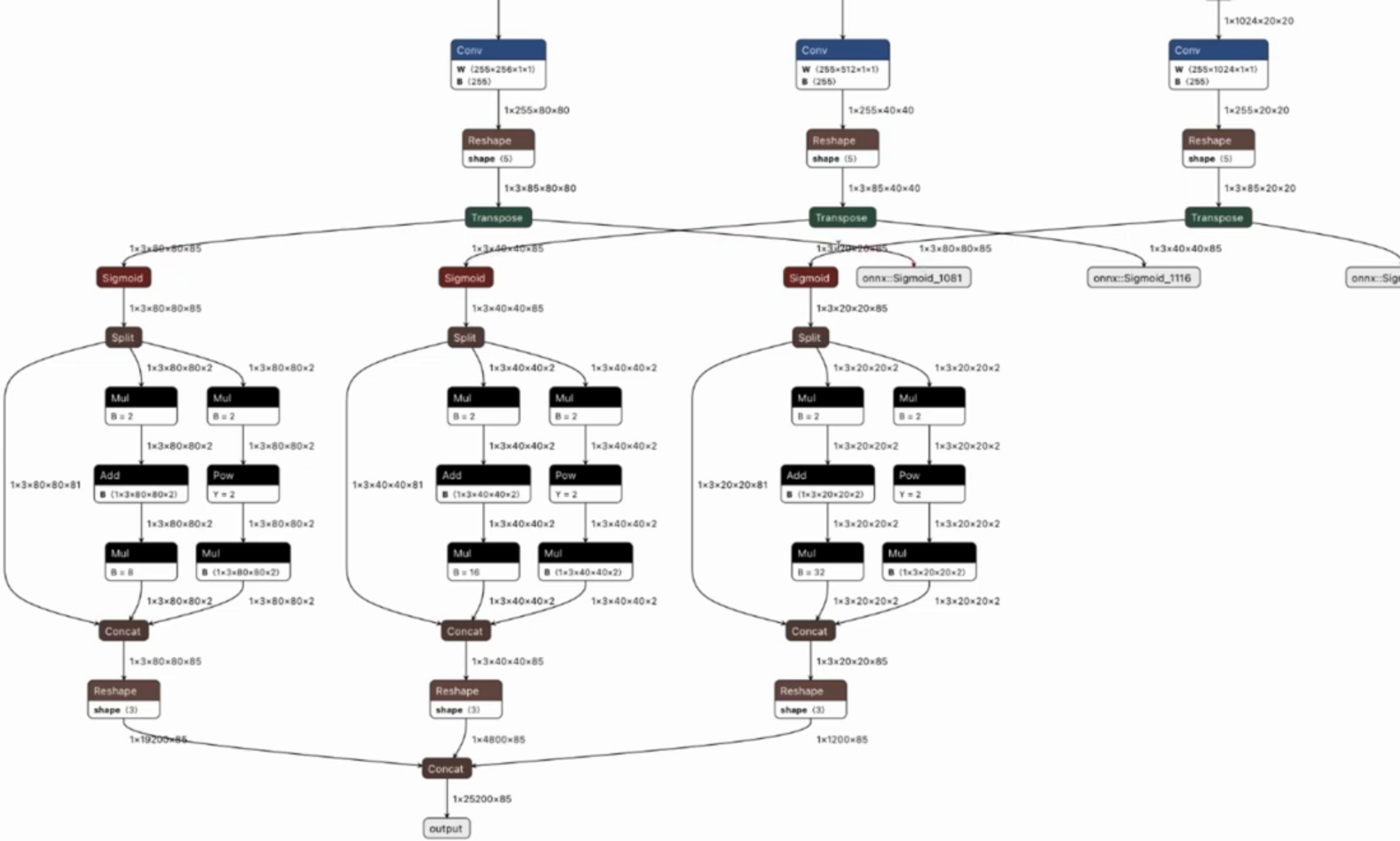
master* Launchpad 0 1 Spaces: 2 UTF-8 LF YAML Indents: 2

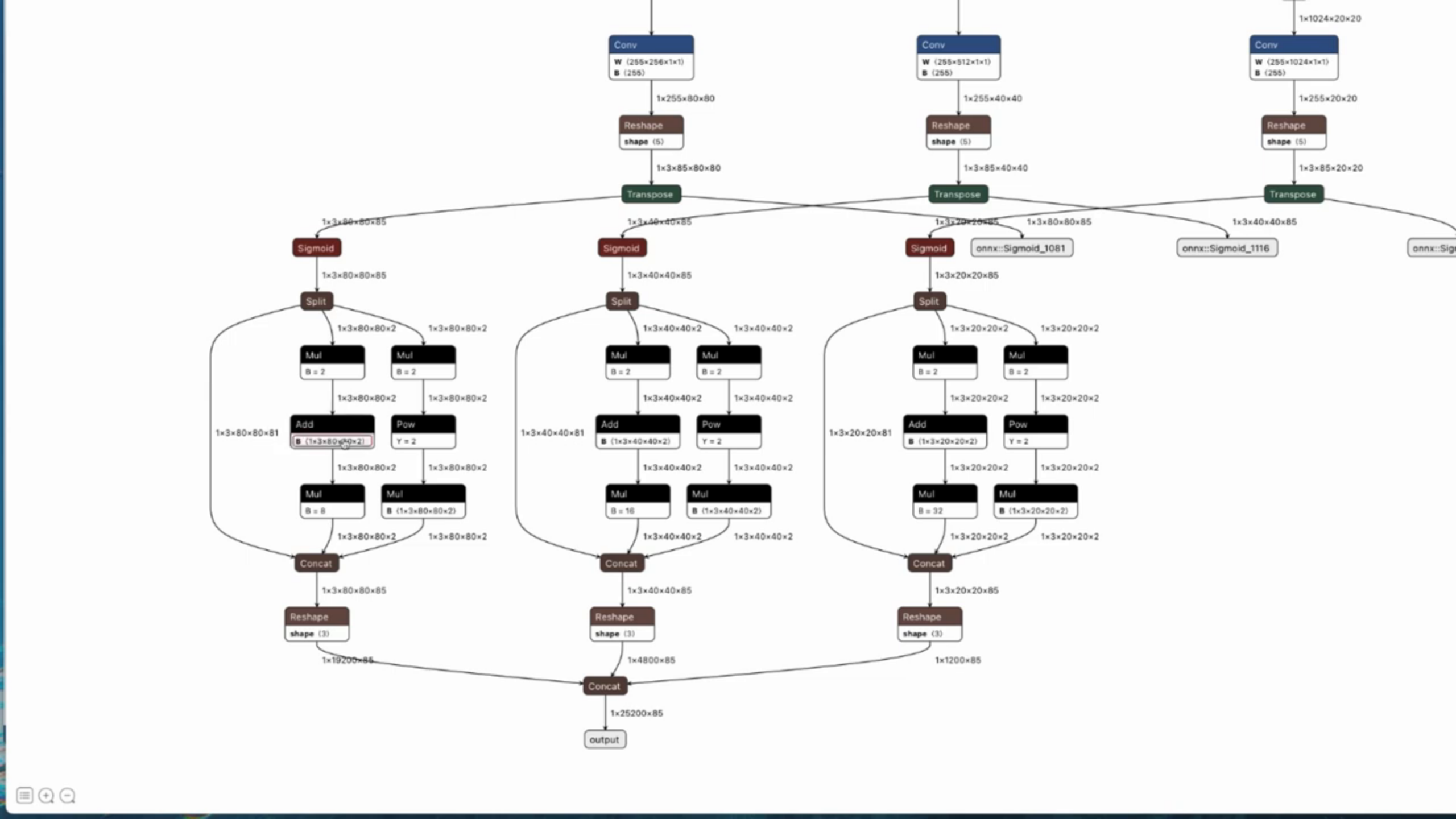


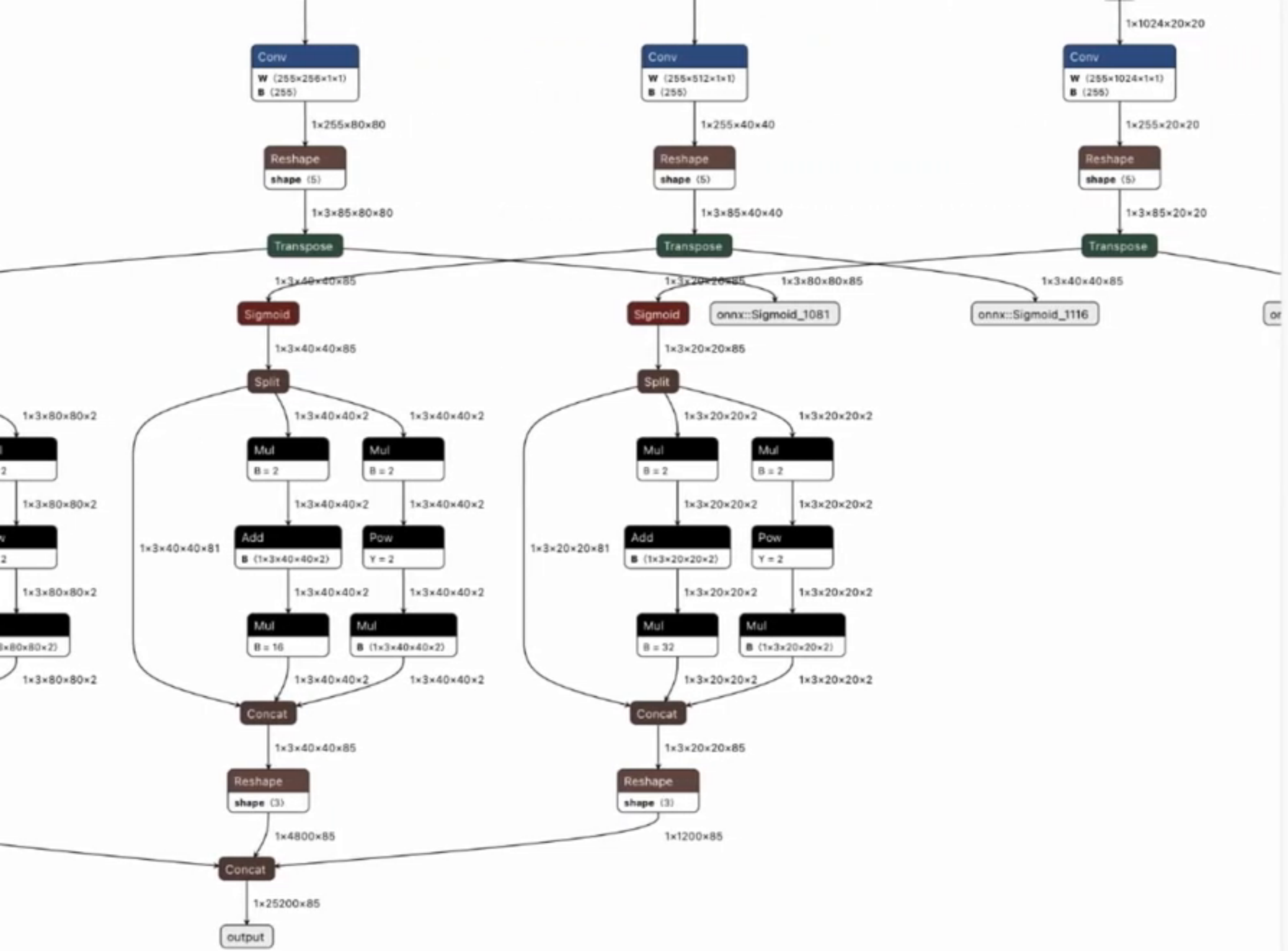

```
40    [-1, 1, Conv, [256, 3, 2]],
41    [[-1, 14], 1, Concat, [1]], # cat head P4
42    [-1, 3, C3, [512, False]], # 20 (P4/16-medium)
```

master* Launchpad 0 1 Spaces: 2 UTF-8 LF YAML Indents: 2

1×256×40×40 Sigmoid 1×256×40×40
1×256×40×40 Mul
1×256×40×40 Concat
1×512×40×40 Conv
W (512×512×1×1) B (512)
1×512×40×40 Sigmoid 1×512×40×40
1×512×40×40 Mul
1×512×40×40 Conv
W (256×512×1×1) B (256)
1×256×40×40 Sigmoid 1×256×40×40
1×256×40×40 Mul
Sigmoid 1×128×80×80
1×128×80×80


Conv
W (255×256×1×1)
B (255)
1×255×80×80
Reshape
shape (5)
1×3×85×80×80
Transpose
1×3×80×80×85
Sigmoid
1×3×80×80×85
Split
1×3×80×80×2
1×3×80×80×2
Mul
B = 2
Mul
B = 2
1×3×80×80×2
1×3×80×80×2
1×3×80×80×81
Add
B (1×3×80×80×2)
Pow
Y = 2
1×3×80×80×2
1×3×80×80×2
Mul
B = 8
Mul
B (1×3×80×80×2)
1×3×80×80×2
1×3×80×80×2
Concat
1×3×80×80×85
Reshape
shape (3)
1×19200×85
Conv
W (255×512×1×1)
B (255)
1×255×40×40
Reshape
shape (5)
1×3×85×40×40
Transpose
1×3×40×40×85
Sigmoid
1×3×40×40×85
Split
1×3×40×40×2
1×3×40×40×2
Mul
B = 2
Mul
B = 2
1×3×40×40×2
1×3×40×40×2
1×3×40×40×81
Add
B (1×3×40×40×2)
Pow
Y = 2
1×3×40×40×2
1×3×40×40×2
Mul
B = 16
Mul
B (1×3×40×40×2)
1×3×40×40×2
1×3×40×40×2
Concat
1×3×40×40×85
Reshape
shape (3)
1×4800×85
Conv
W (255×1024×1×1)
B (255)
1×1024×20×20
1×255×20×20
Reshape
shape (5)
1×3×85×20×20
Transpose
1×3×20×20×85
Sigmoid
1×3×20×20×85
Split
1×3×20×20×2
1×3×20×20×2
Mul
B = 2
Mul
B = 2
1×3×20×20×2
1×3×20×20×2
1×3×20×20×81
Add
B (1×3×20×20×2)
Pow
Y = 2
1×3×20×20×2
1×3×20×20×2
Mul
B = 32
Mul
B (1×3×20×20×2)
1×3×20×20×2
1×3×20×20×2
Concat
1×3×20×20×85
Reshape
shape (3)
1×1200×85
1×3×80×80×85
onnx::Sigmoid_1081
1×3×40×40×85
onnx::Sigmoid_1116
Concat
1×25200×85
output


Conv
W (255×256×1×1)
B (255)
1×255×80×80
Reshape
shape (5)
1×3×85×80×80
Transpose
1×3×80×80×85
Sigmoid
1×3×80×80×85
Split
1×3×80×80×2
1×3×80×80×2
Mul
B = 2
Mul
B = 2
1×3×80×80×2
1×3×80×80×2
1×3×80×80×81
Add
B (1×3×80×80×2)
Pow
Y = 2
1×3×80×80×2
1×3×80×80×2
Mul
B = 8
Mul
B (1×3×80×80×2)
1×3×80×80×2
1×3×80×80×2
Concat
1×3×80×80×85
Reshape
shape (3)
1×19200×85
Conv
W (255×512×1×1)
B (255)
1×255×40×40
Reshape
shape (5)
1×3×85×40×40
Transpose
1×3×40×40×85
Sigmoid
1×3×40×40×85
Split
1×3×40×40×2
1×3×40×40×2
Mul
B = 2
Mul
B = 2
1×3×40×40×2
1×3×40×40×2
1×3×40×40×81
Add
B (1×3×40×40×2)
Pow
Y = 2
1×3×40×40×2
1×3×40×40×2
Mul
B = 16
Mul
B (1×3×40×40×2)
1×3×40×40×2
1×3×40×40×2
Concat
1×3×40×40×85
Reshape
shape (3)
1×4800×85
Conv
W (255×1024×1×1)
B (255)
1×1024×20×20
1×255×20×20
Reshape
shape (5)
1×3×85×20×20
Transpose
1×3×20×20×85
Sigmoid
1×3×20×20×85
Split
1×3×20×20×2
1×3×20×20×2
Mul
B = 2
Mul
B = 2
1×3×20×20×2
1×3×20×20×2
1×3×20×20×81
Add
B (1×3×20×20×2)
Pow
Y = 2
1×3×20×20×2
1×3×20×20×2
Mul
B = 32
Mul
B (1×3×20×20×2)
1×3×20×20×2
1×3×20×20×2
Concat
1×3×20×20×85
Reshape
shape (3)
1×1200×85
1×3×80×80×85
onnx::Sigmoid_1081
1×3×40×40×85
onnx::Sigmoid_1116
Concat
1×25200×85
output

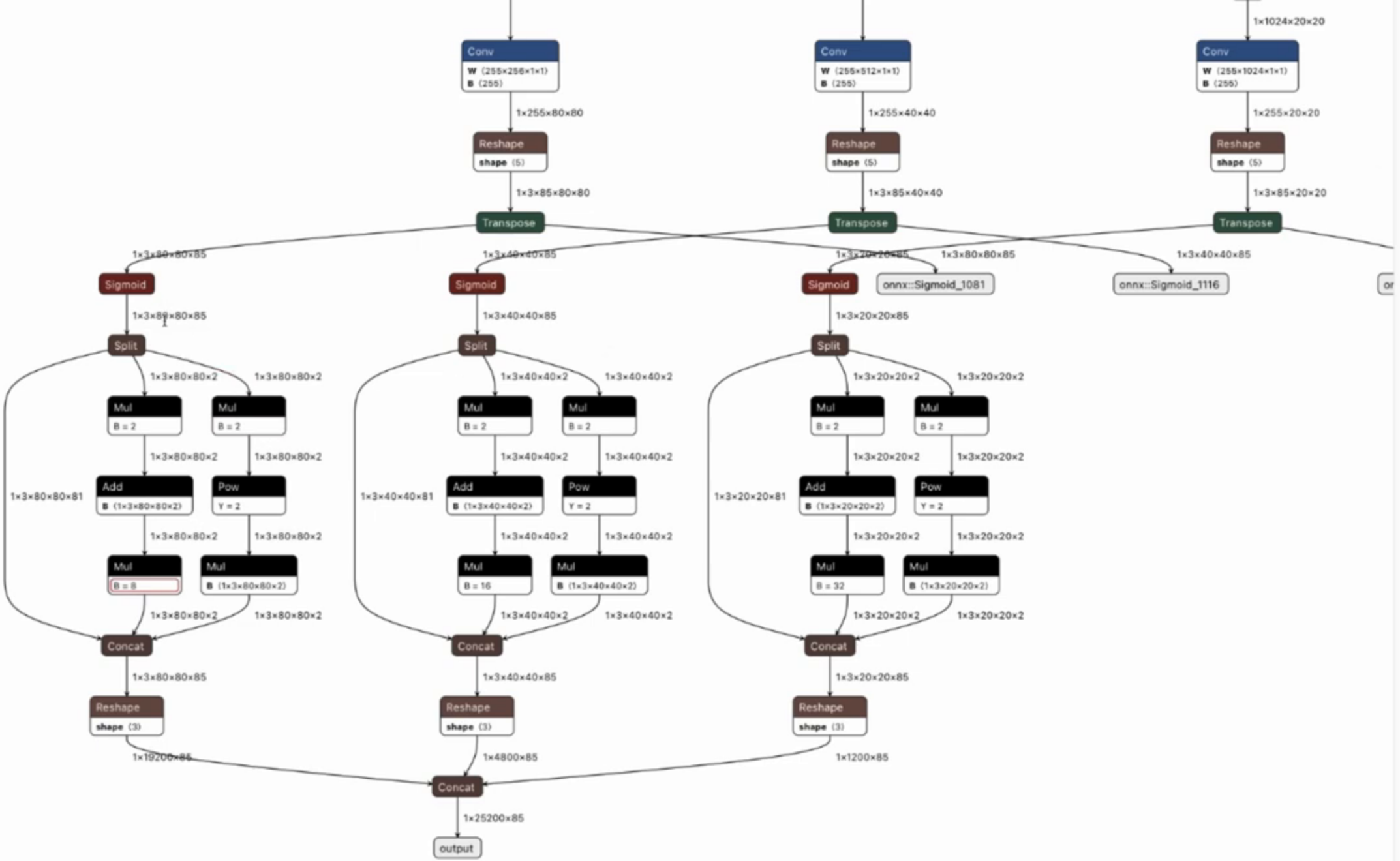


type float32

shape 1,3,80,80,2

value


```
[
    [
        [
            [
                [
                    -0.5,
                    -0.5
                ],
                [
                    0.5,
                    -0.5
                ],
                [
                    1.5,
                    -0.5
                ],
                [
                    2.5,
                    -0.5
                ],
                [
                    3.5,
                    -0.5
                ],
                [
                    4.5,
                    -0.5
                ],
                [
                    5.5,
                    -0.5
                ],
                [
                    6.5,
                    -0.5
                ],
                [
                    7.5,
                    -0.5
                ],
                [
                    8.5,
                    -0.5
                ],
                [
                    9.5,
                    -0.5
                ],
                [
                    10.5,
                    -0.5
                ],
                [
                    11.5,
                    -0.5
                ],
                [
                    12.5,
                    -0.5
                ],
                [
                    13.5,
                    -0.5
                ],
                [
                    14.5,
                    -0.5
                ],
```

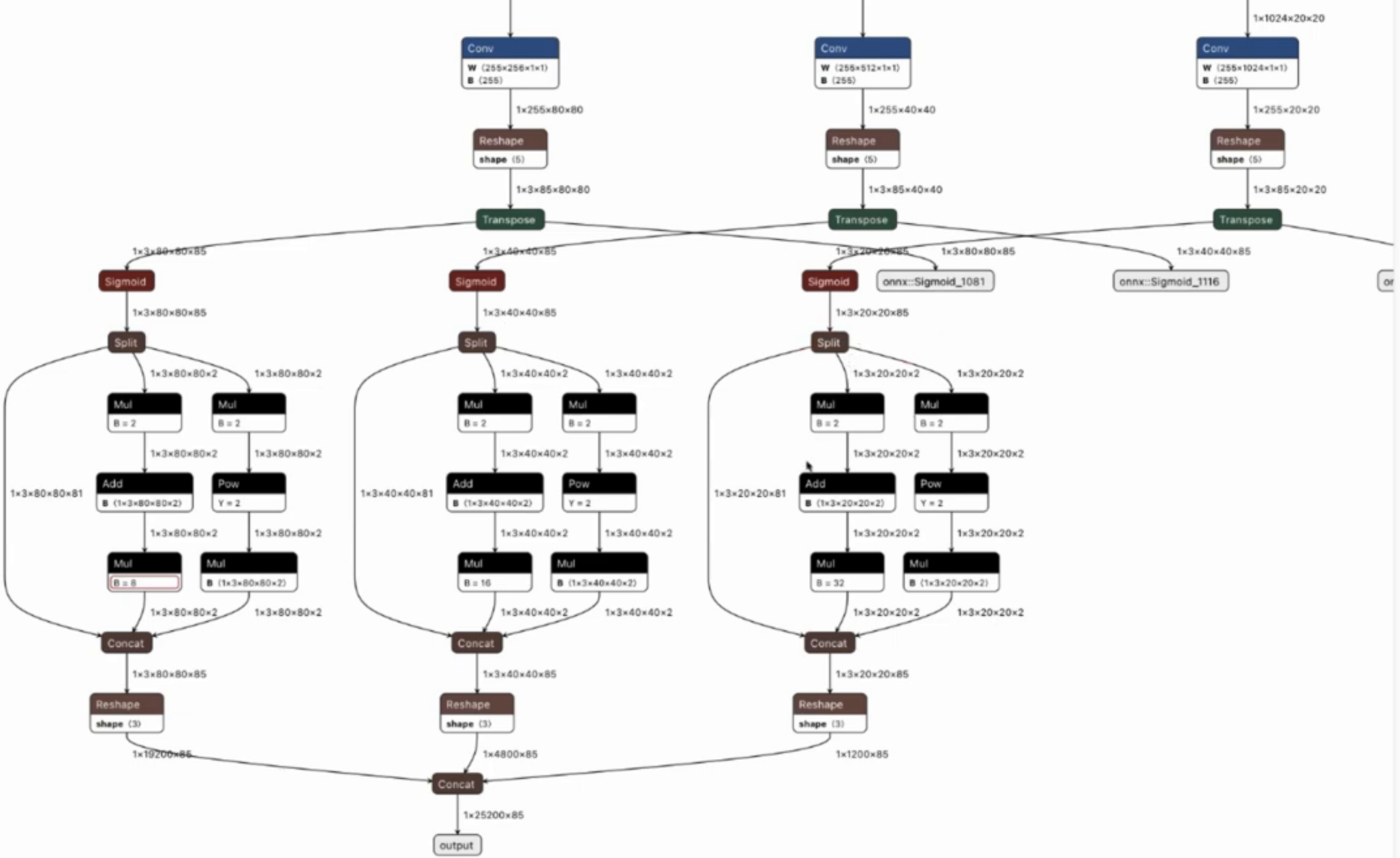


type float3

shape

METRICS

min	8
max	8
mean	8
std	0
sparsity	0.0%

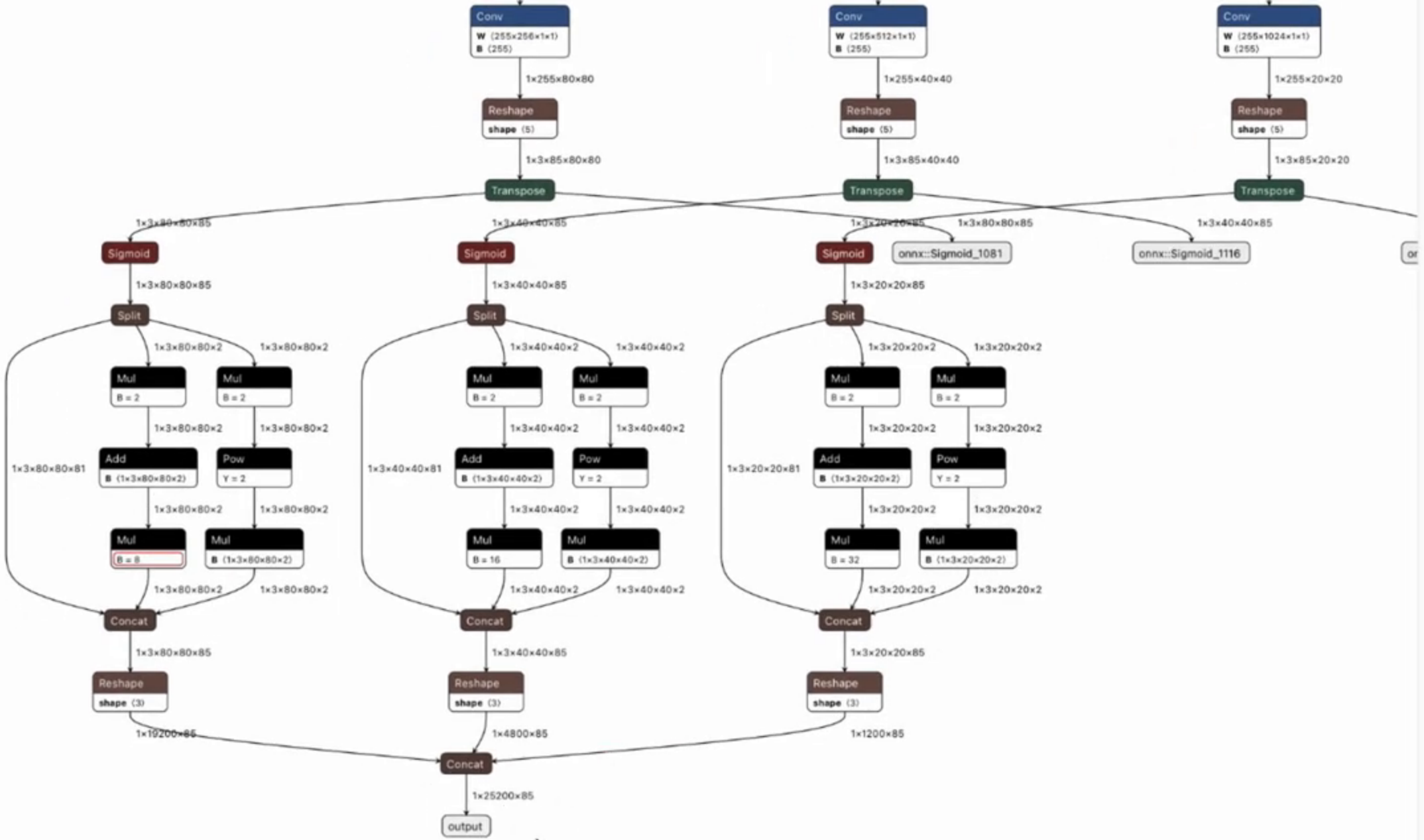


type float3

shape

METRICS

min	8
max	8
mean	8
std	0
sparsity	0.0%

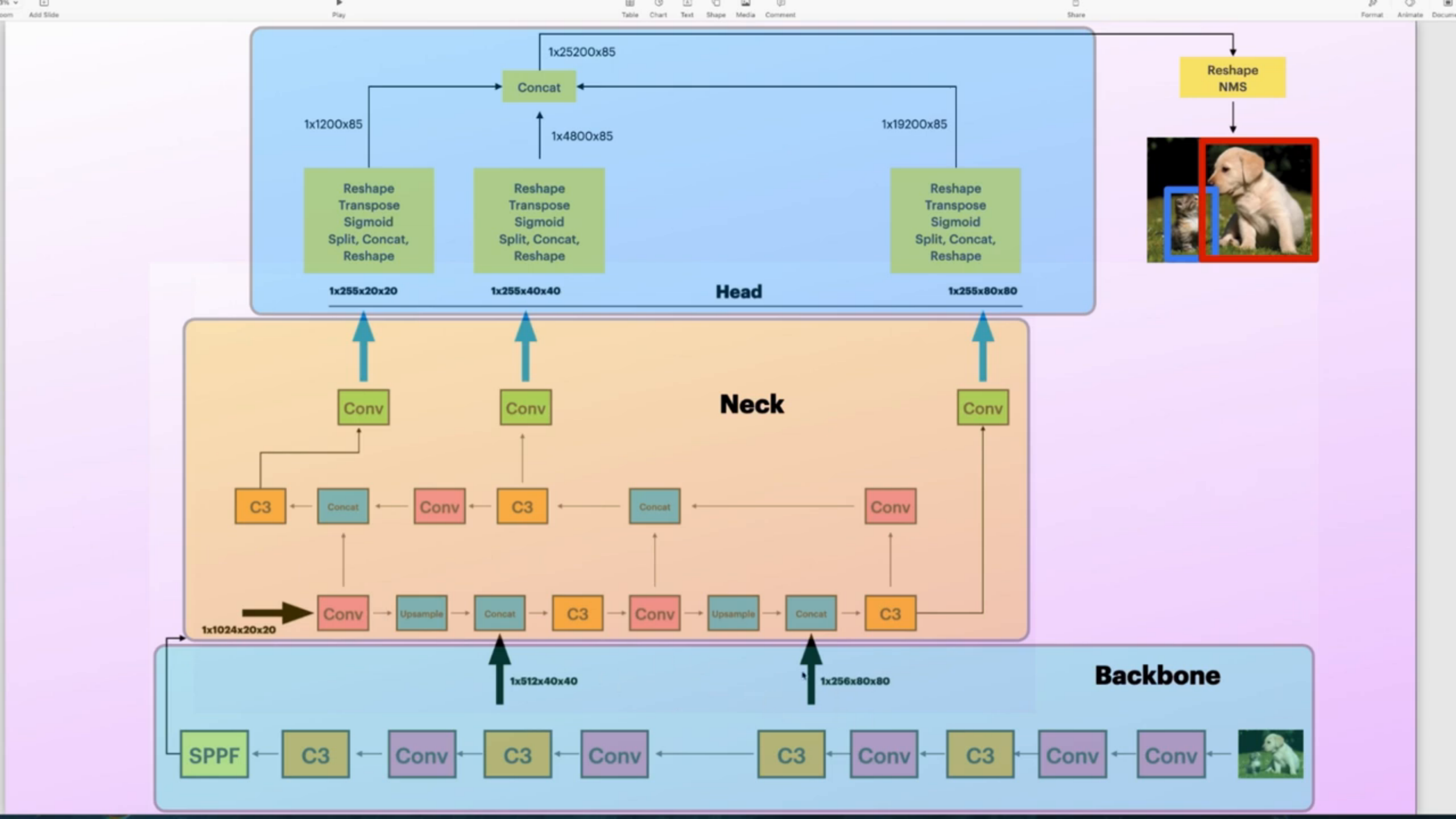


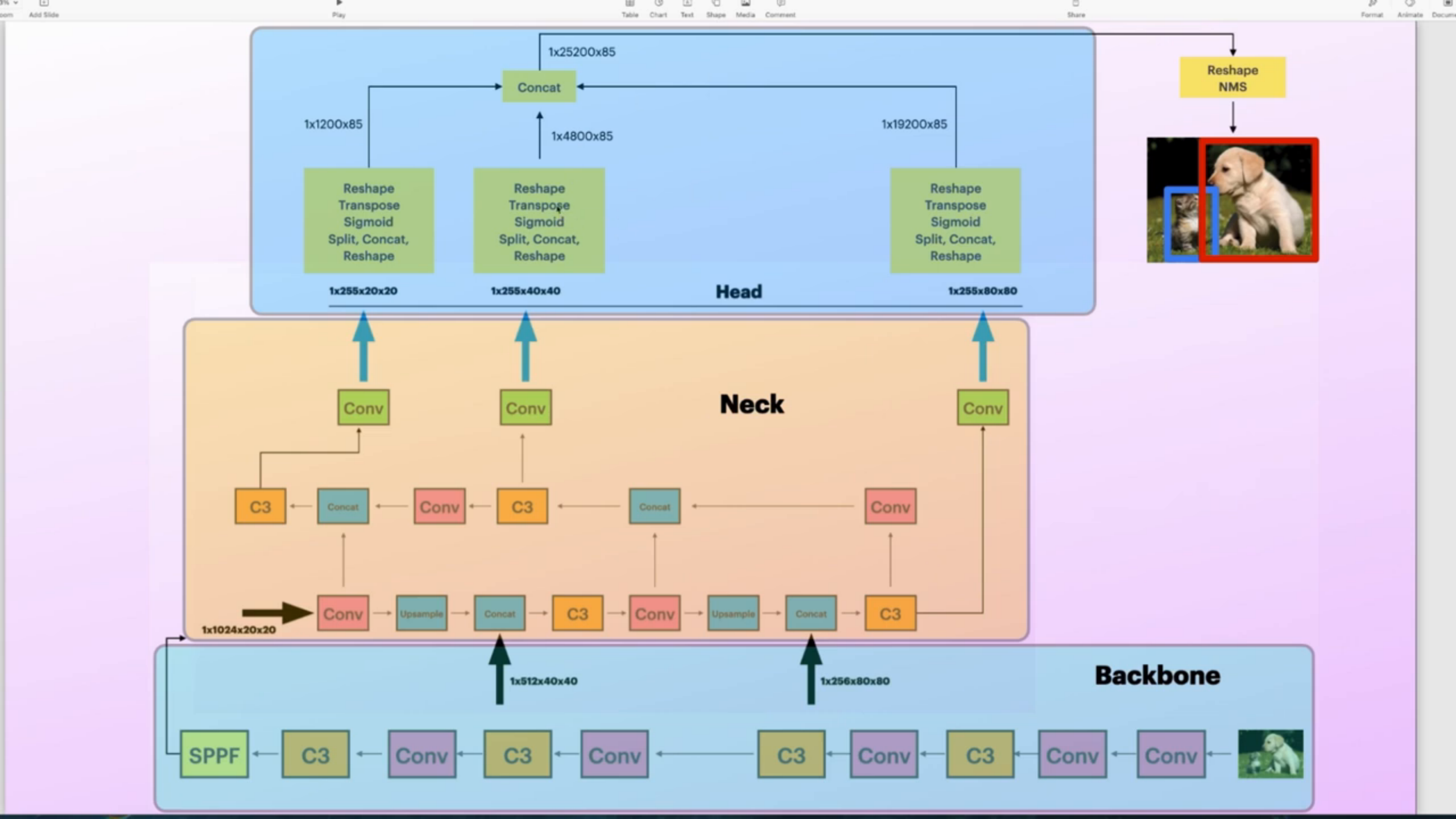
type float3

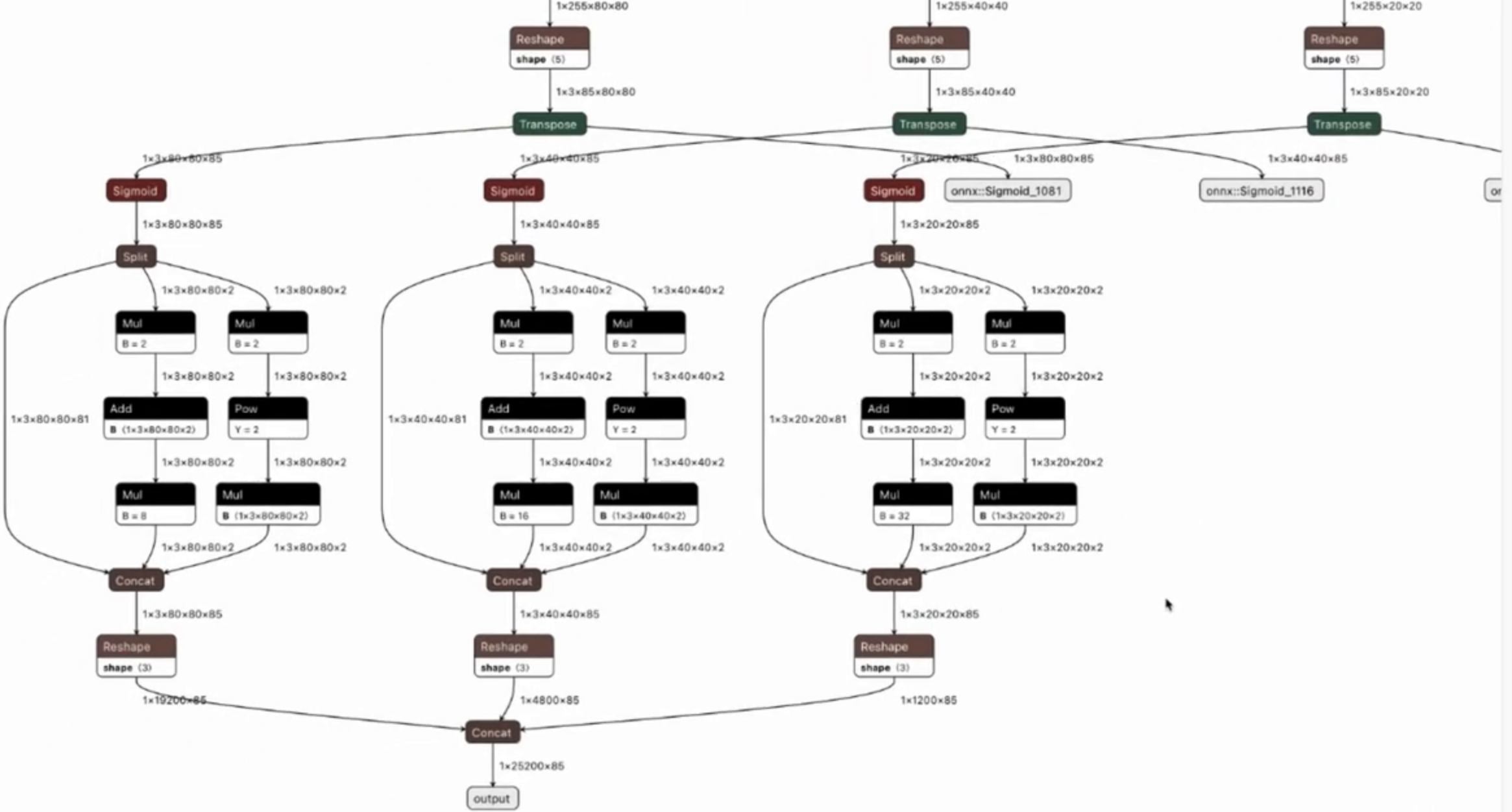
shape

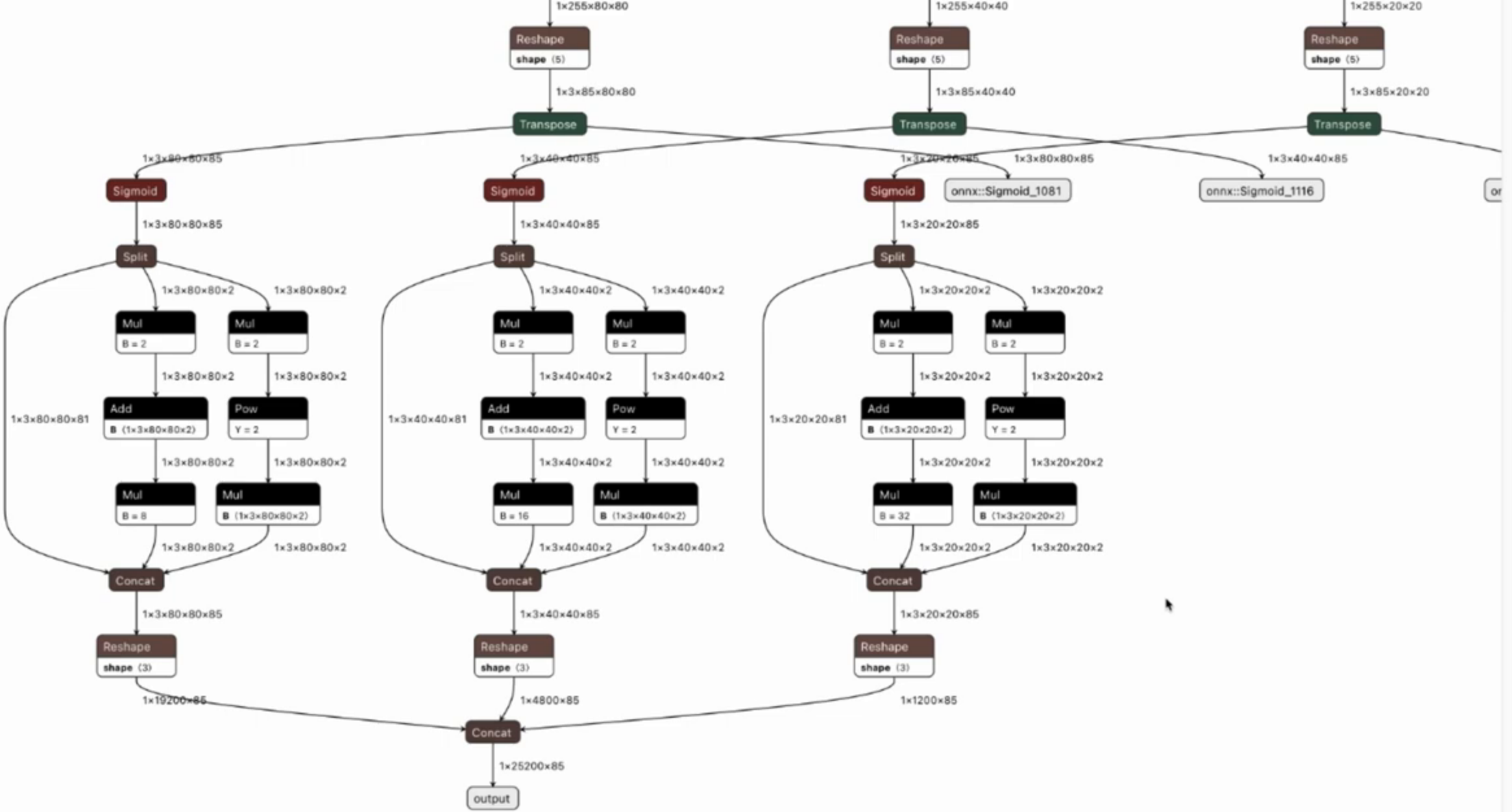
METRICS

min	8
max	8
mean	8
std	0
sparsity	0.0%


1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
1x255x20x20
1x255x40x40
Head
1x255x80x80
Reshape
NMS
Conv
Conv
Neck
Conv
C3
Concat
Conv
C3
Concat
Conv
Conv
Upsample
Concat
C3
Conv
Upsample
Concat
C3
1x1024x20x20
1x512x40x40
1x256x80x80
Backbone
SPPF
C3
Conv
C3
Conv
C3
Conv
C3
Conv
Conv


1x25200x85
Concat
1x1200x85
1x4800x85
1x19200x85
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
Reshape
Transpose
Sigmoid
Split, Concat,
Reshape
1x255x20x20
1x255x40x40
Head
1x255x80x80
Reshape
NMS
Conv
Conv
Neck
Conv
C3
Concat
Conv
C3
Concat
Conv
Conv
Upsample
Concat
C3
Conv
Upsample
Concat
C3
1x1024x20x20
1x512x40x40
1x256x80x80
Backbone
SPPF
C3
Conv
C3
Conv
C3
Conv
C3
Conv
Conv


1×255×80×80
Reshape
shape (5)
1×3×85×80×80
Transpose
1×3×80×80×85
Sigmoid
1×3×80×80×85
Split
1×3×80×80×2
1×3×80×80×2
Mul
B = 2
Mul
B = 2
1×3×80×80×2
1×3×80×80×2
1×3×80×80×81
Add
B (1×3×80×80×2)
Pow
Y = 2
1×3×80×80×2
1×3×80×80×2
Mul
B = 8
Mul
B (1×3×80×80×2)
1×3×80×80×2
1×3×80×80×2
Concat
1×3×80×80×85
Reshape
shape (3)
1×19200×85
1×255×40×40
Reshape
shape (5)
1×3×85×40×40
Transpose
1×3×40×40×85
Sigmoid
1×3×40×40×85
Split
1×3×40×40×2
1×3×40×40×2
Mul
B = 2
Mul
B = 2
1×3×40×40×2
1×3×40×40×2
1×3×40×40×81
Add
B (1×3×40×40×2)
Pow
Y = 2
1×3×40×40×2
1×3×40×40×2
Mul
B = 16
Mul
B (1×3×40×40×2)
1×3×40×40×2
1×3×40×40×2
Concat
1×3×40×40×85
Reshape
shape (3)
1×4800×85
1×255×20×20
Reshape
shape (5)
1×3×85×20×20
Transpose
1×3×20×20×85
Sigmoid
1×3×20×20×85
Split
1×3×20×20×2
1×3×20×20×2
Mul
B = 2
Mul
B = 2
1×3×20×20×2
1×3×20×20×2
1×3×20×20×81
Add
B (1×3×20×20×2)
Pow
Y = 2
1×3×20×20×2
1×3×20×20×2
Mul
B = 32
Mul
B (1×3×20×20×2)
1×3×20×20×2
1×3×20×20×2
Concat
1×3×20×20×85
Reshape
shape (3)
1×1200×85
1×3×80×80×85
onnx::Sigmoid_1081
1×3×40×40×85
onnx::Sigmoid_1116
Concat
1×25200×85
output


1×255×80×80
Reshape
shape (5)
1×3×85×80×80
Transpose
1×3×80×80×85
Sigmoid
1×3×80×80×85
Split
1×3×80×80×2
1×3×80×80×2
Mul
B = 2
Mul
B = 2
1×3×80×80×2
1×3×80×80×2
1×3×80×80×81
Add
B (1×3×80×80×2)
Pow
Y = 2
1×3×80×80×2
1×3×80×80×2
Mul
B = 8
Mul
B (1×3×80×80×2)
1×3×80×80×2
1×3×80×80×2
Concat
1×3×80×80×85
Reshape
shape (3)
1×19200×85
1×255×40×40
Reshape
shape (5)
1×3×85×40×40
Transpose
1×3×40×40×85
Sigmoid
1×3×40×40×85
Split
1×3×40×40×2
1×3×40×40×2
Mul
B = 2
Mul
B = 2
1×3×40×40×2
1×3×40×40×2
1×3×40×40×81
Add
B (1×3×40×40×2)
Pow
Y = 2
1×3×40×40×2
1×3×40×40×2
Mul
B = 16
Mul
B (1×3×40×40×2)
1×3×40×40×2
1×3×40×40×2
Concat
1×3×40×40×85
Reshape
shape (3)
1×4800×85
1×255×20×20
Reshape
shape (5)
1×3×85×20×20
Transpose
1×3×20×20×85
Sigmoid
1×3×20×20×85
Split
1×3×20×20×2
1×3×20×20×2
Mul
B = 2
Mul
B = 2
1×3×20×20×2
1×3×20×20×2
1×3×20×20×81
Add
B (1×3×20×20×2)
Pow
Y = 2
1×3×20×20×2
1×3×20×20×2
Mul
B = 32
Mul
B (1×3×20×20×2)
1×3×20×20×2
1×3×20×20×2
Concat
1×3×20×20×85
Reshape
shape (3)
1×1200×85
1×3×80×80×85
onnx::Sigmoid_1081
1×3×40×40×85
onnx::Sigmoid_1116
Concat
1×25200×85
output