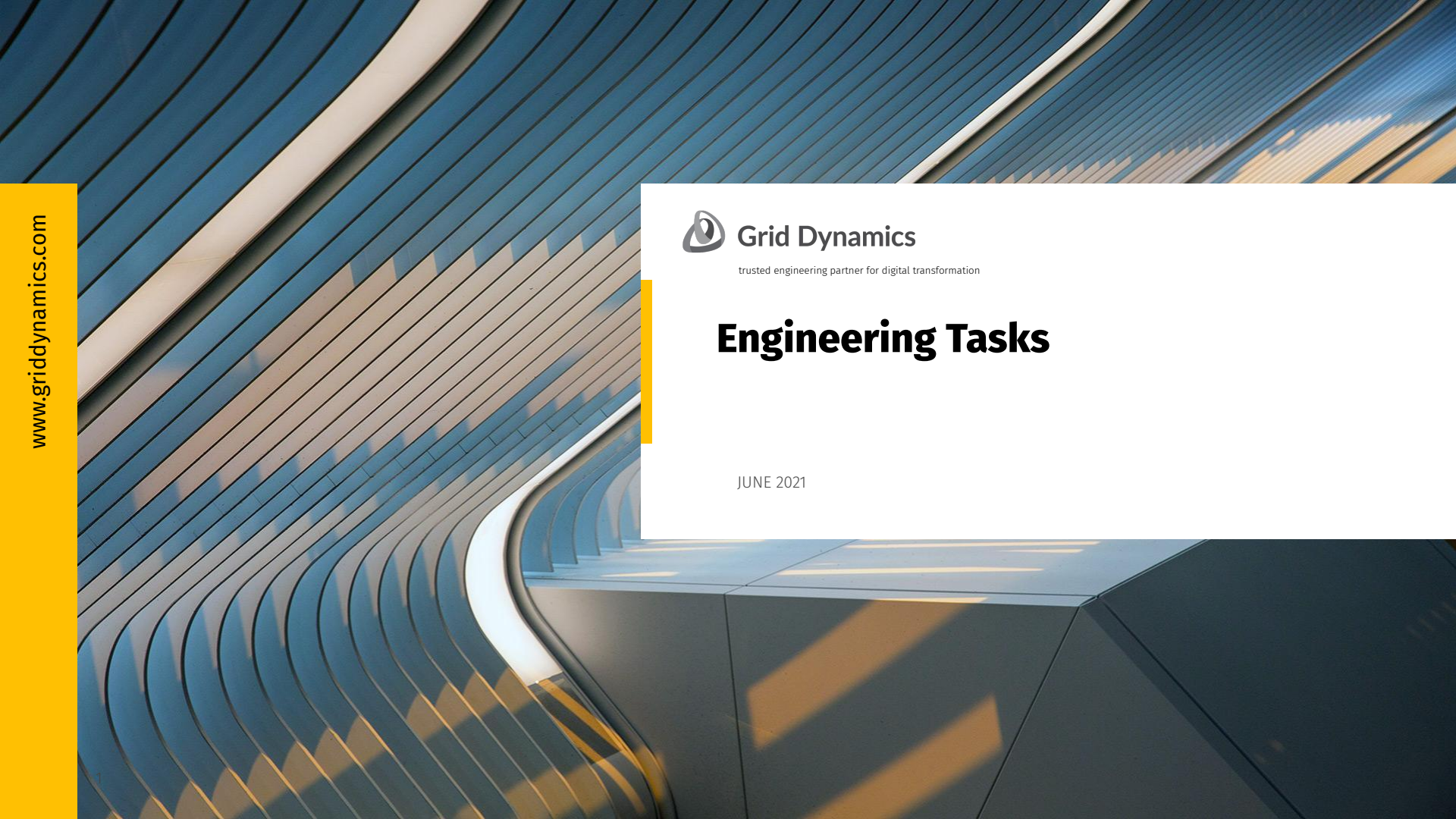


www.griddynamics.com

Grid Dynamics

trusted engineering partner for digital transformation

Engineering Tasks

JUNE 2021

YOLO Pruning

1. **Baseline Model Performance:** Our baseline YOLO model for keypoint detection had mAP50: 81.80% and mAP50-95: 37.99%.

```
metrics.pose.map, metrics.pose.map50
```

```
(0.3799101581327904, 0.8180010357036621)
```

2. **Number of Parameters Pruned:** We have applied **L1 unstructured pruning** and we have pruned 727,606 parameters of convolutional layers out of 3.263 million parameters, i.e, we have 22.297% pruning ratio.

```
total_params, pruned_params, prune_ratio = find_unstructured_prune_ratio(model_2)
total_params, pruned_params, prune_ratio * 100
```

```
(3263104, 727606, 22.297971501980935)
```

3. **Pruned Model Accuracy:** The pruned model accuracy has dropped like mAP50: 79.67% and mAP50-95: 34.83%.

```
[+] mAP50:  0.7967361492523993  mAP50-95:  0.34383602776180106
```

To recover the accuracy lost from pruning, we fine-tune the pruned model(pruning aware training) for **1 epoch** with **lr_intial:1e-7** and **lr_final: 1e-7 * 1e-6, with SGD optimizer.**

```python
model_2.train(data='config.yaml', epochs=1, batch=64, optimizer='SGD', lr0=1e-7, lrf=1e-6, momentum=0.005, imgsz=(640, 480), cache=True, dropout=0, cos_lr=True, close_mosaic=1,
              freeze=21, mosaic=0.0, hsv_h=0.02, hsv_s=0.5, hsv_v=0.4, translate=0.0, scale=0.0, degrees=0)
```

Python

Thus our accuracy increases like mAP50: 80.65% and mAP50-95: 36.34%, but it reduces the pruning ratio from 22.29% to 22.14%.

```python
print("[+] mAP50: ", val_metrics.pose.map50, " mAP50-95: ", val_metrics.pose.map)
```

```
[+] mAP50:  0.8065850026094209  mAP50-95:  0.3634773682006332
```

The pruned model has compression ratio 79.52% as compared to original model that has 90.84%.

```python
compression_ratio_original = (original_compressed_size / original_uncompressed_size) * 100
compression_ratio_pruned = (pruned_compressed_size / pruned_uncompressed_size) * 100
compression_ratio_original, compression_ratio_pruned
```

```
(90.84337086516639, 79.52444319209485)
```

The pruned model compressed file is 12.52% smaller in size as compared to original model compressed size.

```
reduction_in_size = ((original_compressed_size - pruned_compressed_size) / original_compressed_size) * 100
reduction_in_size
```

```
12.520123519460741
```

We ran a test on 500 images of dimensions (480, 640) to calculate the average inference time for pruned and original model. We found that both of them had almost same inference time, this is because we need special hardware/software to leverage the sparse matrix multiplication.

```
print("[+] Original model avg inference time: ", avg_original_speed, " pruned_model inference time: ", avg_pruned_speed)
```

```
[+] Original model avg inference time:  [67.3525903224945]  pruned_model inference time:  [67.78613018989563]
```

YOLO Quantisation

For quantising the yolo model we first convert it to ‘onnx’ format and then quantise it to UInt8.

```
    print("[+] Total Conv layers: ", total_count, " quantized layers: ", quantized_count)
5]
```

```
[+] Total Conv layers:  159.0  quantized layers:  73.0
```

The accuracy of pruned and quantised model has dropped to mAP50: 79.85% mAP50-95: 37.06%.

```
    print("[+] Pruned but not quantized model mAP50: ", val_metrics.pose.map50, " mAP50-95: ", val_metrics.pose.map)
]
```

```
[+] Pruned but not quantized model mAP50:  0.7985162750263713  mAP50-95:  0.3706975042217301
```

The pruned and quantised model is even more compressible than pruned and original model.

```
    compression_ratio_original = (orginal_onnx_compressed / orginal_onnx_uncompressed) * 100
    compression_ratio_pruned = (pruned_onnx_compressed / pruned_onnx_uncompressed) * 100
    compression_ratio_pruned_and_quantized = (pruned_and_quantized_onnx_compressed / pruned_and_quantized_onnx_uncompressed) * 100
    compression_ratio_original, compression_ratio_pruned, compression_ratio_pruned_and_quantized
```

```
(85.38681090962497, 71.57060038280643, 61.321472072432215)
```

Finally with pruning and quantisation we achieve 79.99% reduction in compressed file size as compared to original model.

```
final_size_reduction_percentage = ((orginal_onnx_compressed - pruned_and_quantized_onnx_compressed) / orginal_onnx_compressed) * 100
final_size_reduction_percentage
```

```
79.99725393313773
```

Since we did dynamic quantisation and dynamic quantisation calculates the quantisation parameters dynamically, which leads to an increase in inference time from 53.63ms to 78.73ms.

```
avg_original_speed, avg_pruned_and_quantized_speed
```

```
([53.632475614547 73], [78.730140209198])
```

Pose Classification Model Pruning

We first convert our tensorflow model to ‘onnx’ format and from ‘onnx’ format to ‘torch’ model.

The baseline classification model has classification accuracy of 61.24%.

```
print(f"[+] Accuracy of baseline model: {accuracy}")
```

```
[+] Accuracy of baseline model: 61.24314442413162
```

The we apply **L1 unstructured** pruning to the model and pruned 2.185 million parameters out of 5.784 million parameters, which is a pruning percent of 37.77%.

```
total_params, pruned_params, prune_ratio = find_unstructured_prune_ratio(pose_classification_pytorch_model)
print("[+] Total Params: ", total_params, " number of weights pruned: ", pruned_params, " percent pruned: ", prune_ratio * 100)
```

```
[+] Total Params:  5784828  number of weights pruned:  2185495  percent pruned:  37.779774956143896
```

Then we evaluate the accuracy of the pruned model, and its classification accuracy has dropped to 50.36%.

```
print(f"[+] Accuracy of pruned model: {accuracy}")
```

```
[+] Accuracy of pruned model: 50.36563071297989
```

Then we fine tune the pruned model again using **‘Adam’ optimiser** and **learning_rate=1e-5 for 2 epochs.**

Thus we recover our accuracy from 50.36% to 61.60% while maintaining the same pruning ratio.

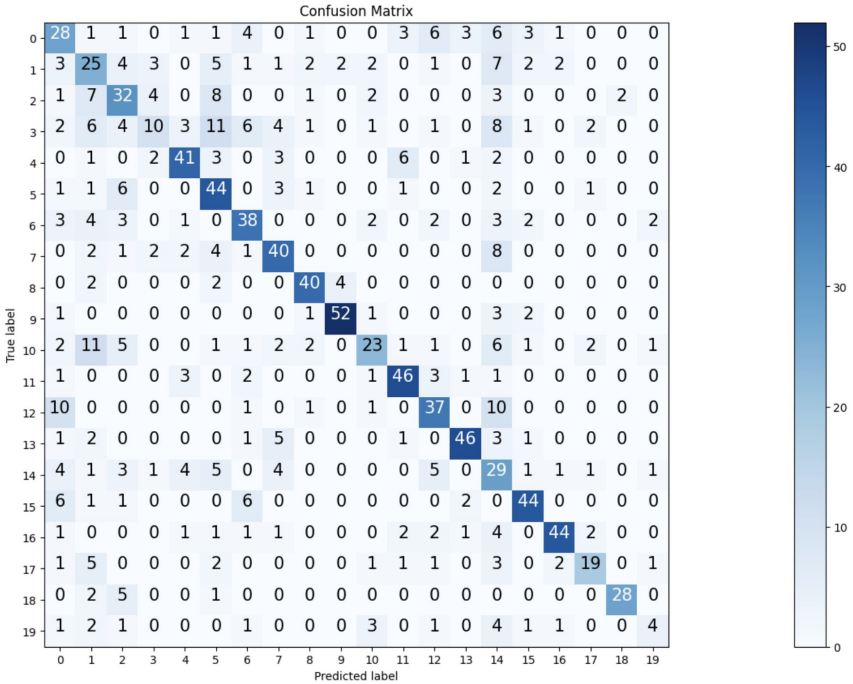
```
    print("[+] Accuracy of fine-tuned pruned model: ", accuracy)
```

```
[+] Accuracy of fine-tuned pruned model:  61.608775137111515
```

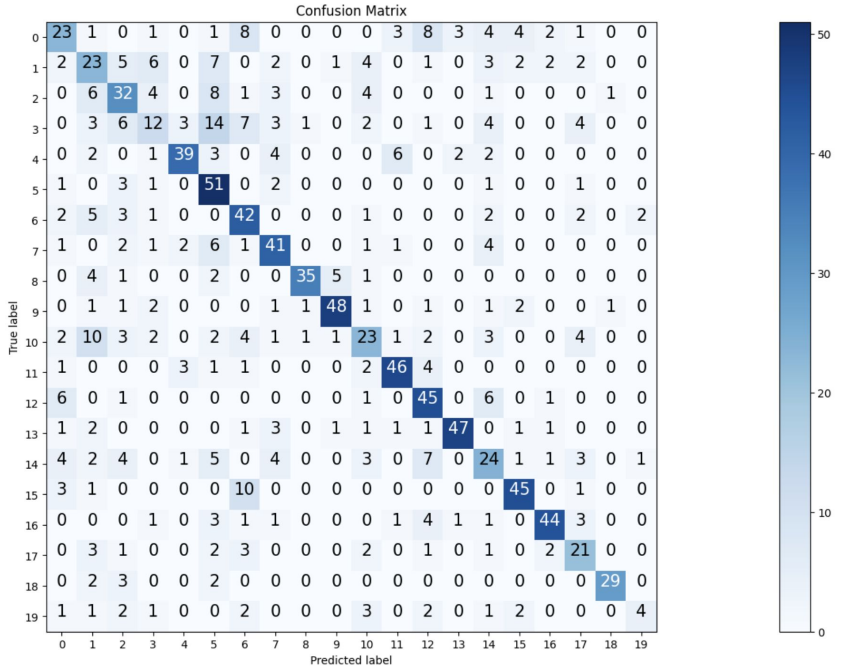
```
total_params, pruned_params, prune_ratio = find_unstructured_prune_ratio(pose_model)
print("[+] Total Params: ", total_params, " number of weights pruned: ", pruned_params, " percent pruned: ", prune_ratio * 100)
```

```
[+] Total Params:  5784828  number of weights pruned:  2185495  percent pruned:  37.779774956143896
```

Below is a side by side comparison of the confusion matrix of original and pruned models.



Original Model



Final Pruned Model

Class_Names = {

0: ‘sports’,

1: ‘miscellaneous’,

2: ‘home activities’,

3: ‘occupation’,

4: ‘fishing and hunting’,

5: ‘home repair’,

6: ‘conditioning exercise’,

7: ‘lawn and garden’,

8: ‘religious activities’,

9: ‘music playing’,

10: ‘inactivity quiet/light’,

11: ‘water activities’,

12: ‘running’,

13: ‘winter activities’

14: ‘walking’,

15: ‘dancing’,

16: ‘bicycling’,

17: ‘transportation’,

18: ‘self care’,

19: ‘volunteer activities’