TCL

# How to win **MAI'2022 Depth Estimation** challenge: manual backbone optimization combined with smart model pruning


authors:

Łukasz Treszczotko

Maciej Smyl

TCL RESEARCH EU, 05 NOVEMBER 2022

# TCL Research EU

## Key Facts

- ✔ TCL is one of the largest consumer electronics producers in the world
- ✔ 2nd largest TV producer in the world
- ✔ 35 R&D centers around the world
- ✔ Operates in 160+ countries
- ✔ Over 75 000 employees
- ✔ **TCL Research EU** in Warsaw since 2018


# AutoML Team


- ✔ Neural Architecture Search
- ✔ Automated model compression
- ✔ and more…

- ✔ Optimize models for on-device deployments (smartphones, TVs, …)
- ✔ **Mobile AI challenges on an everyday basis**



- ✔ Creating internal TCL tools for automated ML

# Efficient deep neural networks - more challenging than it seems

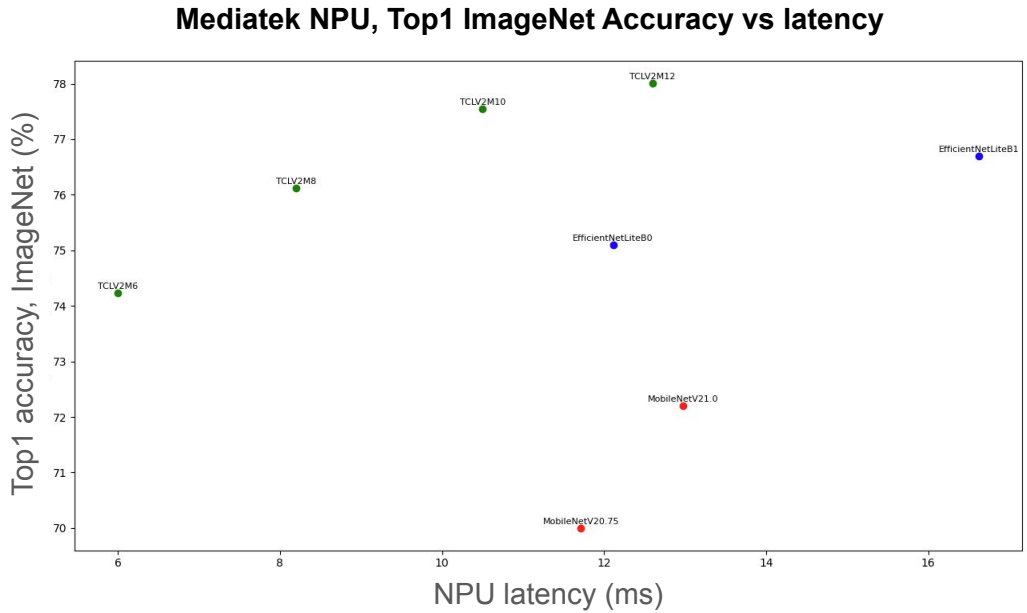
towards efficient dnn models

Towards efficient models:

- ➔ AI systems included in more and more **type of devices**
- ➔ Integrated AI chips
- ➔ Increasing demand on **real-time applications**

Efficiency estimation:

- **Parameters count**
- **Flops** (Multi-Adds)




| | MobileNetV2 1.0 | EfficientNet Lite B0 | **TCLV2M8** |
|---|---|---|---|
| Flops | 609 M | 777 M | **1.1 G** |
| Params | 3.5 M | 4.6 M | **14.2 M** |
| Latency | 13 ms | 12.1 | **8.2 ms** |
| ImageNet Top1 Accuracy | 72.2 % | 75.1 % | **76.1 %** |

# Simplified Unet Architecture For Fast Depth Estimation

fast and accurate

TCL

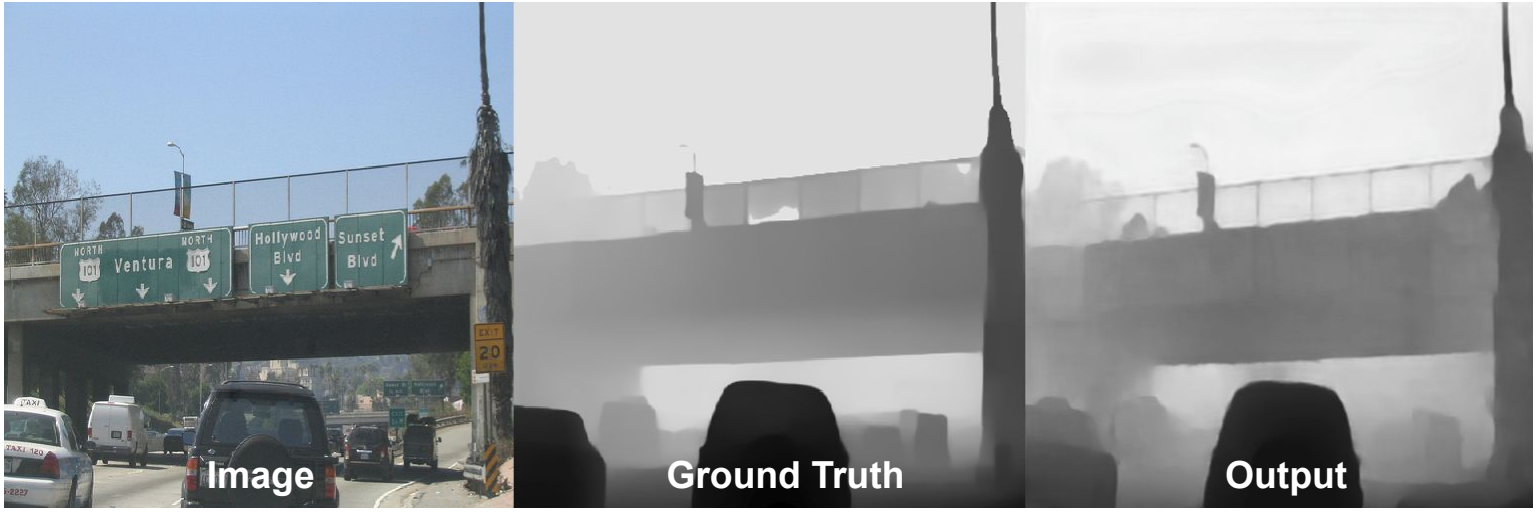
| Team | Author | Framework | Model Size, MB | si-RMSE↓ | RMSE↓ | LOG10↓ | REL↓ | Runtime, ms ↓ | Final Score |
|---|---|---|---|---|---|---|---|---|---|
| TCL | TCL | TensorFlow | 2.9 | **0.2773** | **3.47** | **0.1103** | 0.2997 | 46 | **298** |
| AIIA HIT | Zhenyu Li | PyTorch / TensorFlow | 1.5 | 0.311 | 3.79 | 0.1241 | 0.3427 | **37** | 232 |
| MiAIgo | ChaoMI | PyTorch / TensorFlow | 1.0 | 0.299 | 3.89 | 0.1349 | 0.3807 | 54 | 188 |
| Tencent GY-Lab | Parkzyzhang | PyTorch / TensorFlow | 3.4 | 0.303 | 3.8 | 0.1899 | 0.3014 | 68 | 141 |
| Tencent GY-Lab* | Parkzyzhang | PyTorch / TensorFlow | 3.4 | 0.2836 | 3.56 | 0.1121 | **0.2690** | 103 | 122 |
| SmartLab | RocheL | TensorFlow | 7.1 | 0.3296 | 4.06 | 0.1378 | 0.3662 | 65 | 102 |
| JMU-CVLab | mvc | PyTorch / TensorFlow | 3.5 | 0.3498 | 4.46 | 0.1402 | 0.3404 | 139 | 36 |
| ICL | Byung Hyun Lee | PyTorch / TensorFlow | 5.9 | 0.338 | 6.73 | 0.3323 | 0.5070 | 142 | 42 |

30%

# Monocular Depth Estimation Challenge: intro

Hardware-awareness, channel pruning and expert knowledge

- Optimize the trade-off between latency and quality
- The quality of the model was given by scale-invariant RMSE metric
- Latency was measured on **Raspberry Pi 4** platform
- The training data was provided by the organizers. It constituted of around 7.5K pairs of 480x640 RGB / depth-map pairs.
- The final metric was given by: $\text{Final Score} = \frac{2^{-20 \cdot \text{si-RMSE}}}{C \cdot \text{runtime}}$, where C was some predefined constant



# Monocular Depth Estimation Challenge: our approach

Hardware-awareness, channel pruning and expert knowledge

TCL

- **Set up the device for latency measurement as quickly as possible**

- **Iterate as much as possible**
  - try out different backbone models
  - try out different feature fusion layers
  - measure the latency before training to quickly remove architectures that are obviously too large (no matter their quality)

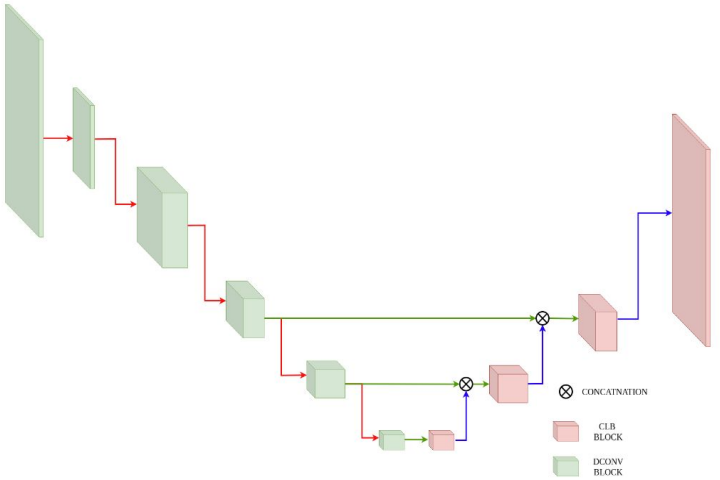
- Once we have a trained model, **run automated channel pruning to compress it and add reparametrization blocks** to enable more expressive training of a really small model.

# Monocular Depth Estimation Challenge: our approach - architecture.

Hardware-awareness, channel pruning and expert knowledge

- The final architecture was rather simple (it must have been given the brutal latency penalty). We used **MobileNetV3 backbone** (after downsampling from 480x640 to 128x160) and a simple feature fusion based on **FPN**.

  At some point we even managed to squeeze in a small vision transformer block into feature fusion which was almost as good as FPNet-like fusion, but for final submission we used FPNet fusion.



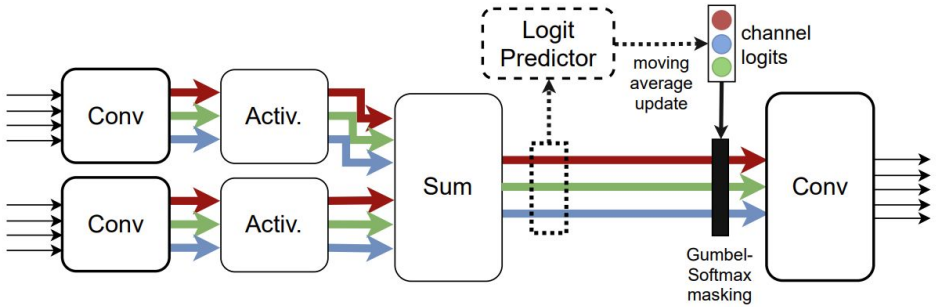
# Monocular Depth Estimation Challenge: model compression.

Hardware-awareness, channel pruning and expert knowledge

- Once the model was trained we run an **internally developed channel pruning algorithm** to make the model even smaller (about 40% FLOPs reduction seemed to be optimal as far as the final score was concerned).

  **Our method:**

  - ➔ works during training, and optimizes jointly the accuracy and the model size
  - ➔ learns channel importance directly from feature maps
  - ➔ automatically detects the interdependent blocks of channels
  - ➔ offers device-aware pruning modes that can be adjusted to specific hardware platforms (e.g. removing groups of N channels, where N can be 8/16/32/…, depending on hardware)



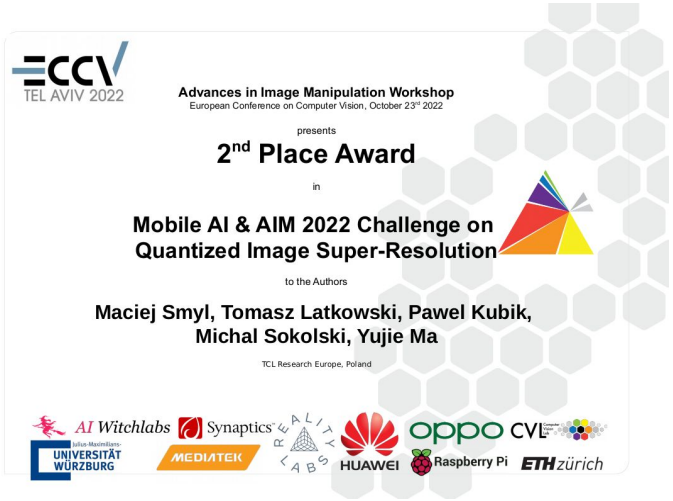

- Just to demonstrate how small the actual model was we can say that given the same input resolution 480x640, MobileNetV2 was > 20 times larger.

# Quantized Image Super Resolution Challenge

## Designing efficient yet powerful model

TCL

| Team | Author | Framework | Model Size, KB | PSNR↑ INT8 | SSIM↑ Model | Δ PSNR Drop, FP32 → INT8 | Runtime, ms ↓ CPU | Runtime, ms ↓ NPU | Speed-Up | Final Score |
|---|---|---|---|---|---|---|---|---|---|---|
| Z6 | ganzoo | Keras / TensorFlow | 67 | 30.03 | 0.8738 | 0.06 | 809 | 19.2 | 42.1 | 22.22 |
| TCLResearchEurope | maciejos_s | Keras / TensorFlow | 53 | 29.88 | 0.8705 | 0.13 | 824 | 15.9 | 51.8 | 21.84 |
| ECNUSR | CCjiahao | Keras /TensorFlow | 50 | 29.82 | 0.8697 | 0.10 | 553 | 15.1 | 36.6 | 21.08 |
| LCVG | zion | Keras / TensorFlow | 48 | 29.76 | 0.8675 | 0.11 | 787 | **15.0** | 52.5 | 19.59 |
| BOE-IOT-AIBD | NBCS | Keras / TensorFlow | 57 | 29.80 | 0.8675 | 0.19 | 877 | 16.1 | 54.5 | 19.27 |
| NJUST | kkkls | TensorFlow | 57 | 29.76 | 0.8676 | 0.16 | 767 | 15.8 | 48.5 | 18.56 |
| Antins_cv | fz | Keras / TensorFlow | 38 | 29.58 | 0.8609 | n.a. | **543** | 15.2 | 35.7 | 15.02 |
| GenMedia Group | stevek | Keras / TensorFlow | 56 | 29.90 | 0.8704 | 0.11 | 855 | 25.6 | 33.4 | 13.91 |
| Vccip | Huaen | PyTorch / TensorFlow | 67 | 29.98 | 0.8729 | n.a. | 1042 | 30.5 | 34.2 | 13.07 |
| MegSR | balabala | Keras / TensorFlow | 65 | 29.94 | 0.8704 | n.a. | 965 | 29.8 | 32.4 | 12.65 |
| DoubleZ | gideon | Keras / TensorFlow | 63 | 29.94 | 0.8712 | 0.1 | 990 | 30.1 | 32.9 | 12.54 |
| Jeremy Kwon | alan_jaeger | TensorFlow | 48 | 29.80 | 0.8691 | 0.09 | 782 | 25.7 | 30.4 | 12.09 |
| Lab216 | sissie | TensorFlow | 78 | 29.94 | 0.8728 | 0 | 1110 | 31.8 | 34.9 | 11.85 |
| TOVB | jklovezhang | Keras / TensorFlow | 56 | 30.01 | 0.8740 | 0.04 | 950 | 43.3 | 21.9 | 9.60 |
| ABPN [19] | baseline | - | 53 | 29.87 | 0.8686 | n.a. | 998 | 36.9 | 27 | 9.27 |
| Samsung Research | xiaozhazha | TensorFlow | 57 | 29.95 | 0.8728 | 0.1 | 941 | 43.2 | 21.8 | 8.84 |
| Rtsisr2022 | rtsisr | Keras / TensorFlow | 68 | 30.00 | 0.8729 | 0.09 | 977 | 46.4 | 21.1 | 8.83 |
| Aselsan Research | deepernewbie | Keras / TensorFlow | 30 | 29.67 | 0.8651 | n.a. | 598 | 30.2 | 19.8 | 8.59 |
| Klab_SR | FykAikawa | TensorFlow | 39 | 29.88 | 0.8700 | n.a. | 850 | 43.1 | 19.7 | 8.05 |
| TCL Research HK | mrblue | Keras / TensorFlow | 121 | **30.10** | **0.8751** | 0.04 | 1772 | 60.2 | 29.4 | 7.81 |
| RepGSR | yilitiaotiaotang | Keras / TensorFlow | 90 | 30.06 | 0.8739 | 0.02 | 1679 | 61.3 | 27.4 | 7.26 |
| ICL | smhong | Keras / TensorFlow | 55 | 29.76 | 0.8641 | 0.23 | 949 | 43.2 | 22.0 | 6.79 |
| Just A try | kia350 | Keras / TensorFlow | 39 | 29.75 | 0.8671 | n.a. | 766 | 42.9 | 17.9 | 6.76 |
| Bilibili AI | Sail | Keras / TensorFlow | 75 | 29.99 | 0.8729 | n.a. | 1075 | 68.2 | 15.8 | 5.92 |
| MobileSR | garasgaras | TensorFlow | 82 | 30.02 | 0.8735 | 0.07 | 1057 | 72.4 | 14.6 | 5.82 |
| A+ regression [72] | Baseline | | | 29.32 | 0.8520 | - | - | - | - | - |
| Bicubic Upscaling | Baseline | | | 28.26 | 0.8277 | - | - | - | - | - |




60 fps, Full HD upsampling

Efficient and Accurate Quantized Image Super-Resolution on Mobile NPUs, Mobile AI & AIM 2022 challenge: Report, *to be published*

# Quantized Image Super Resolution Challenge

Designing efficient yet powerful model

TCL

3x bicubic upsample

Ground Truth

# Quantized Image Super Resolution Challenge

Designing efficient yet powerful model

TCL

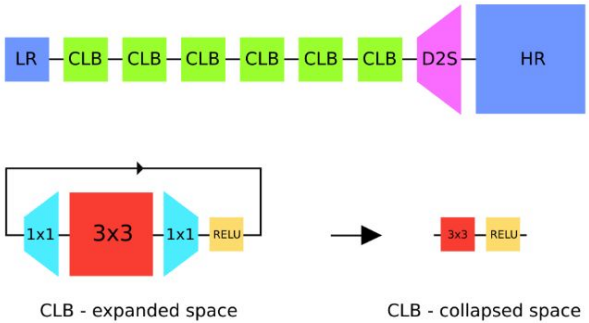
Platform: **Synaptics VS680**

Goal: efficient and high quality **3x upsampling**

How to design efficient and powerful DNN model for **NPU**?

Our solution:

- prunable re-parametrization
- purely convolutional inference graph
- Quantization-aware training for nearly lossless int8 quantization



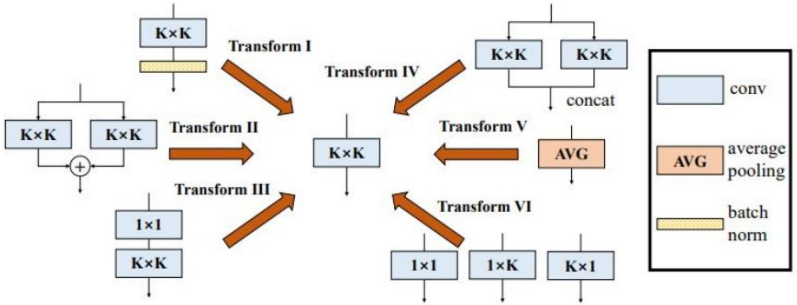
# Structural Re-paramerization

Designing efficient yet powerful model

Convolution linearity:

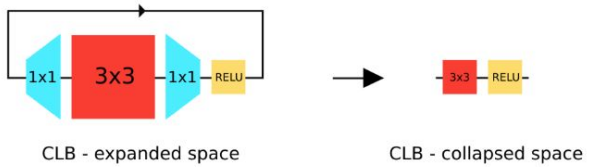
$$\boldsymbol{I} \circledast (p\boldsymbol{F}) = p(\boldsymbol{I} \circledast \boldsymbol{F}), \forall p \in \mathbb{R},$$

$$\boldsymbol{I} \circledast \boldsymbol{F}^{(1)} + \boldsymbol{I} \circledast \boldsymbol{F}^{(2)} = \boldsymbol{I} \circledast (\boldsymbol{F}^{(1)} + \boldsymbol{F}^{(2)})$$




*Diverse Branch Block: Building a Convolution as an Inception-like Unit, Xiaohan Ding*

Sequence of convolutions can be analytically collapsed into **single convolution**
&
Skip connection can be viewed as a **single convolution with identity kernel**




Complex training graph -> **Plain inference structure**

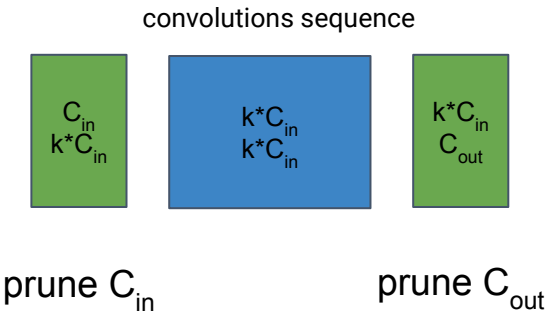
(Almost) free performance boost!

Skip connection:

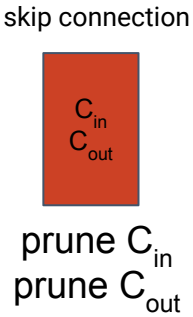
$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

TCL

## Re-parametrization module - channels pruning



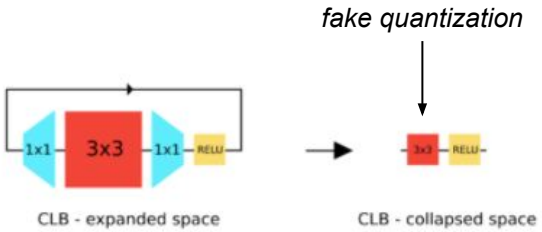

**Shareable** (*over convolutions sequence and skip connection branches*) pruning mask



Quantization in **expanded space**

During **forward pass**:

1. re-parametrize the CLB module weights into single convolutional kernel
2. perform quantization and de-quantization of resulting kernel




CLB - forward pass scheme

# Summary

1. Growing demand on on-device AI application (often real-time) results in increased interest in efficient DNN architectures.

2. In order to design powerful and fast model, target device runtime measurements should be employed.

3. Best results can be obtained with automated NAS methods, but manual tweaks can also work reasonably well.

4. *More = better* - don't be afraid to incorporate several optimization method simultaneously.

# Mobile AI Challenges 2022 - TCL Research EU Team

Łukasz Treszczotko
(DE task leader)

Maciej Smyl
(SR task leader)

Xin Chang

Piotr Książek

Paweł Kubik

Michał Łopuszyński

Tomasz Latkowski

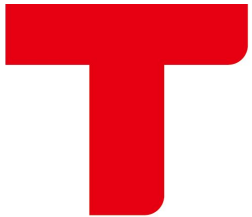
Maciej Pióro

Rafał Rudnicki

Michał Sokólski

Yujie Ma

*Michał Kudelski*
*(managerial support,*
*TCL AutoML Team Leader)*



THANK YOU