SEMANTIC SEGMENTATION ON CAMVID DATASET

Foundations of Deep Learning Project – AA 2023/2024

Cristian Longoni (871070)

Robin Smith (839696)

Sergio Verga (859200)

OUTLINE OF THE PRESENTATION

INTRODUCTION

EXPLORATION AND PRE-PROCESSING

MODEL FROM SCRATCH

PRETRAINED MODELS

RESULTS ANALYSIS

CONCLUSIONS AND FUTURE PERSPECTIVES

INTRODUCTION

Task: **semantic segmentation** on CamVid dataset

Multi-class, single-label

Model from scratch

Architecture inspired by **UNet**

Pretrained models

MobileNetV2

DeepLab

Dataset splitting (701 istances)

401 images for training

150 for validation

150 for test

DATASET EXPLORATION

701 images and labels: **no duplicates** found

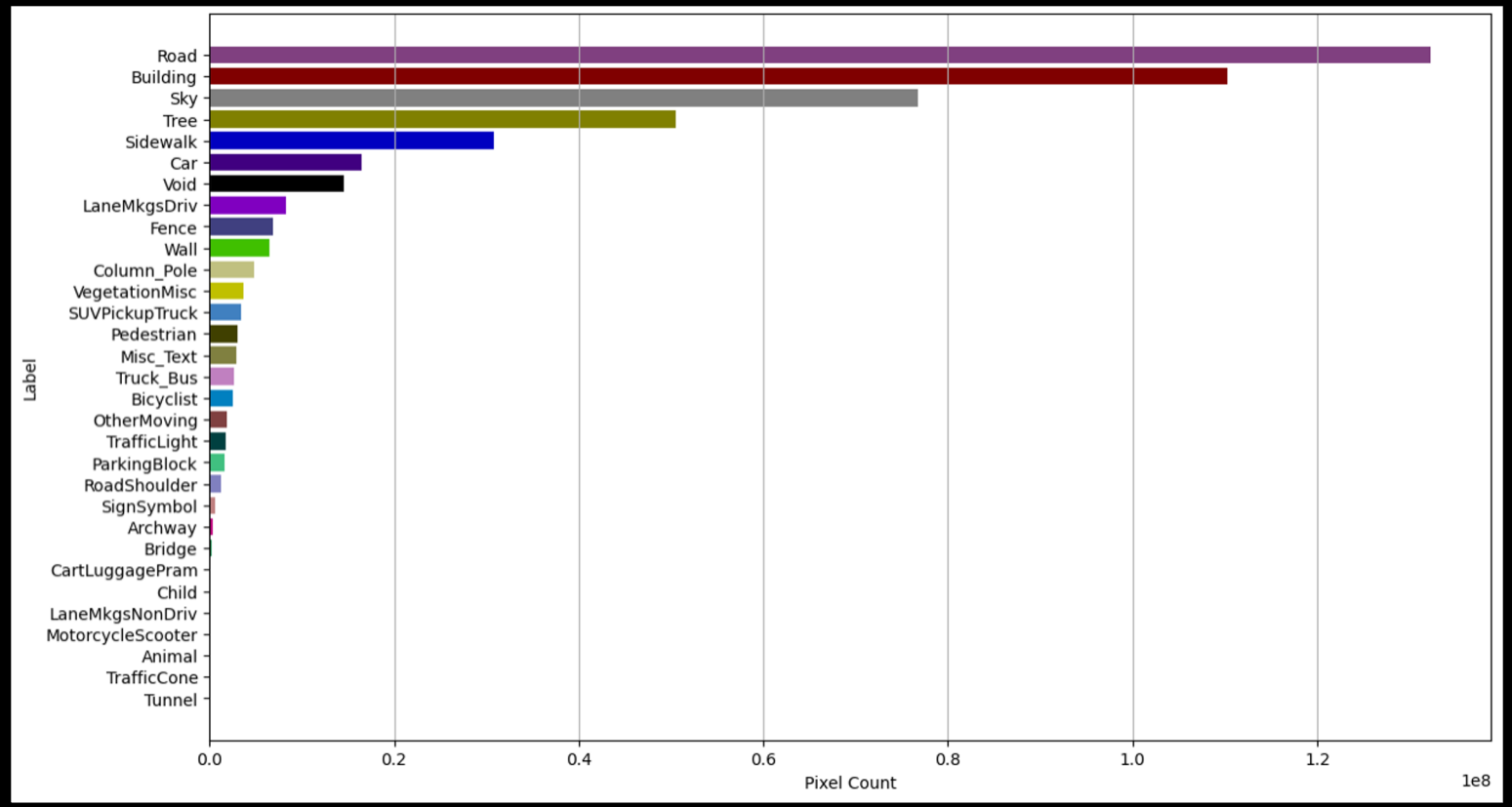
32 semantic classes in total – **Void** class indicates the unlabeled or uncertain pixels

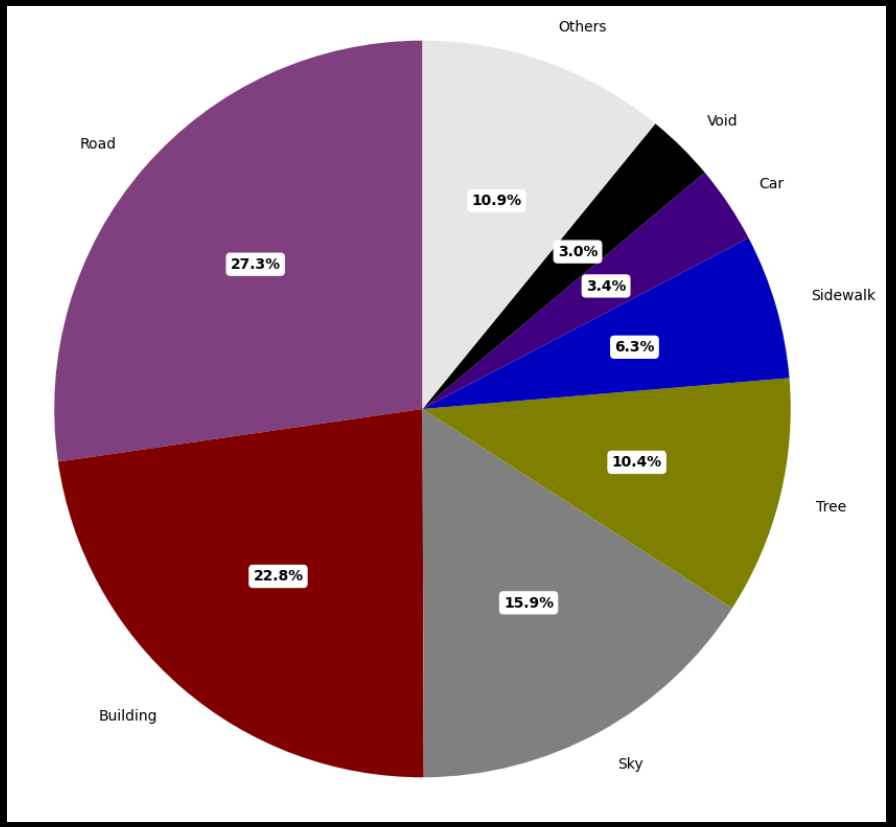
Analysis of **class distribution** of the pixels

Class occurrence computation for each image

DATASET EXPLORATION

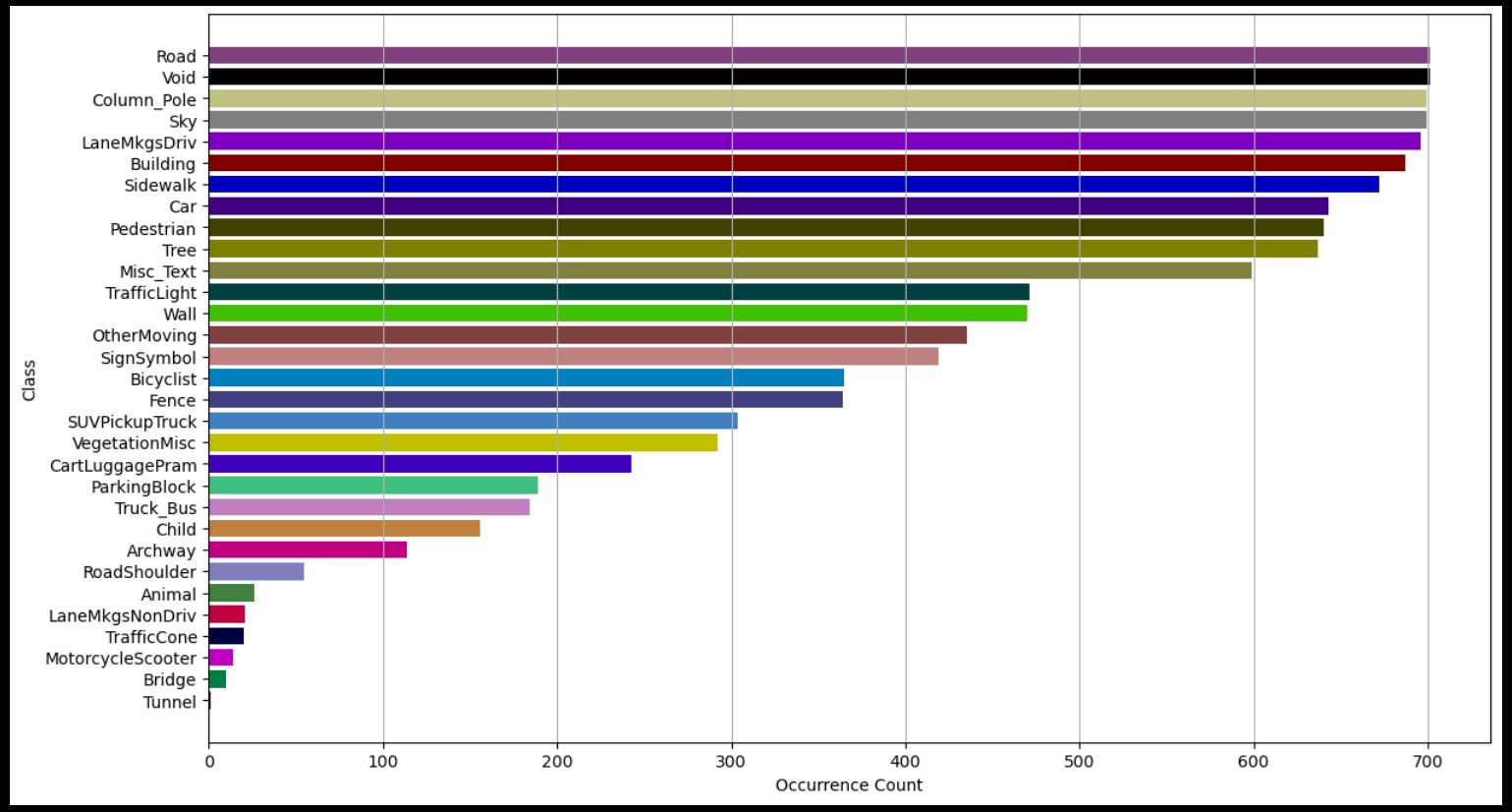
Pixel label counts and percentages within the dataset





DATASET EXPLORATION

Class occurrence plot



Note:

- **Road** and **void** labels appear in all the 701 images
- **Tunnel** label appears in only one image

DATASET PRE-PROCESSING

Shuffle

Shuffling in unison images and labels to **break temporal correlation**.

Resize

Resizing from 960x720 pixels to 224x224 pixels with:

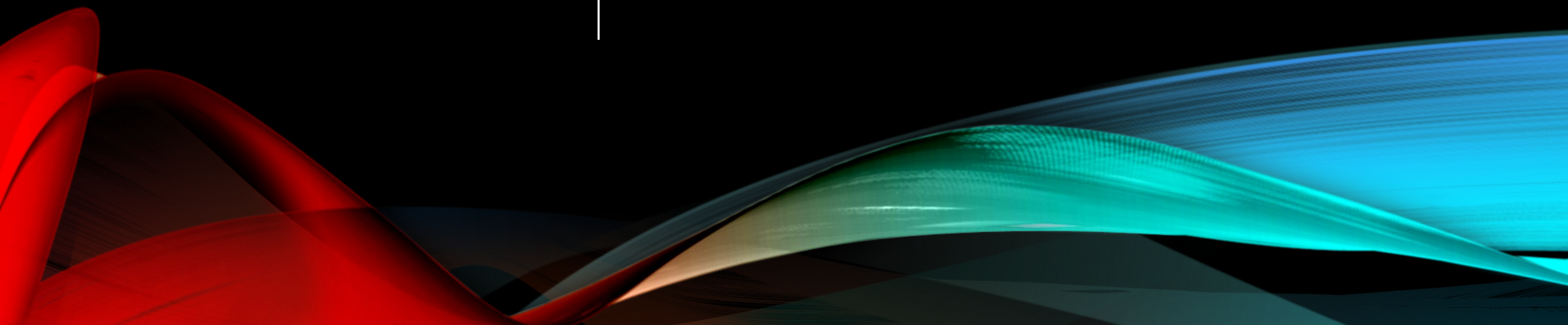
- **Bicubic interpolation** for raw images
- **Nearest neighbor** for labels

Encode

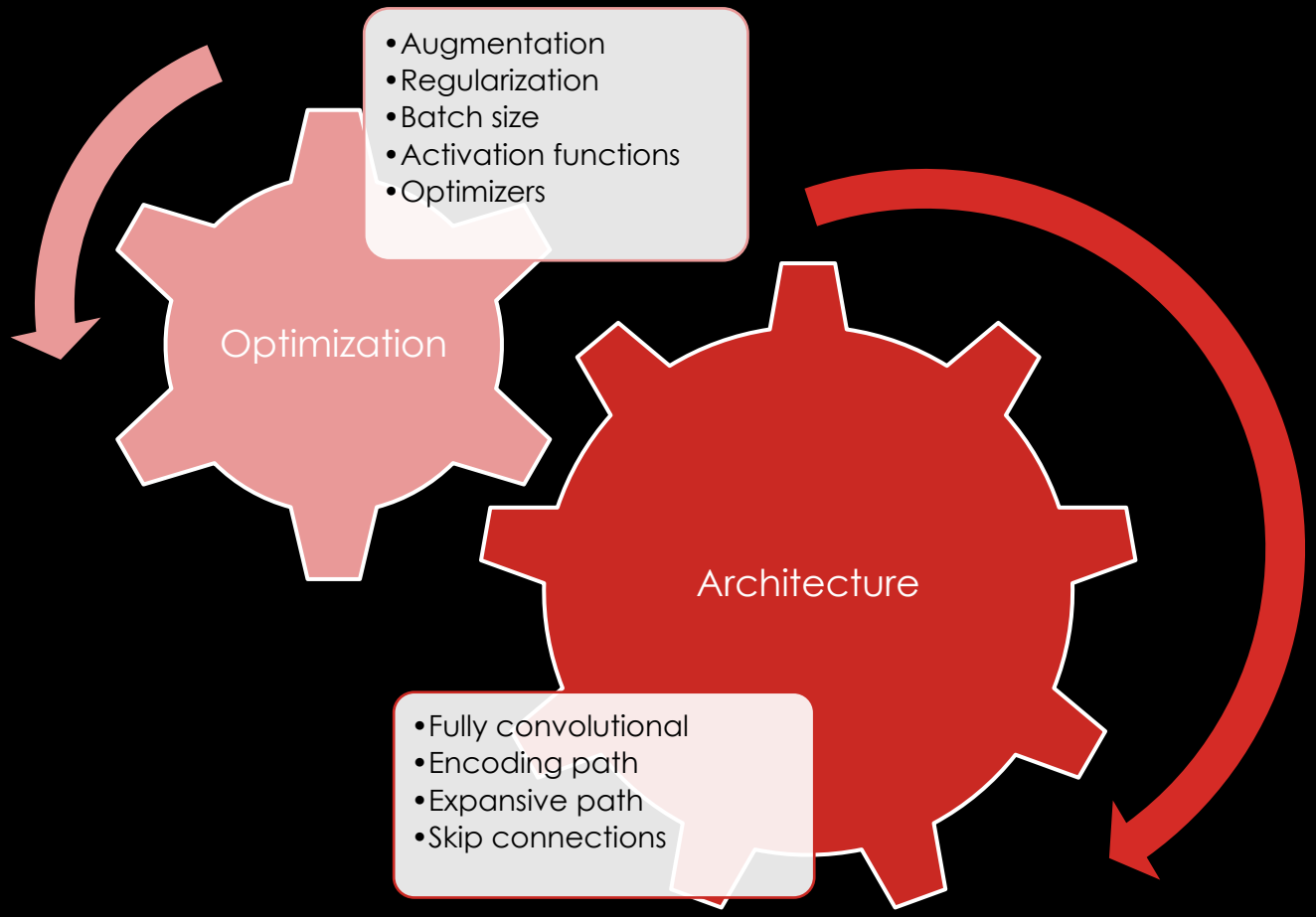
Encoding labels assigning each RGB triple to a **class index**.

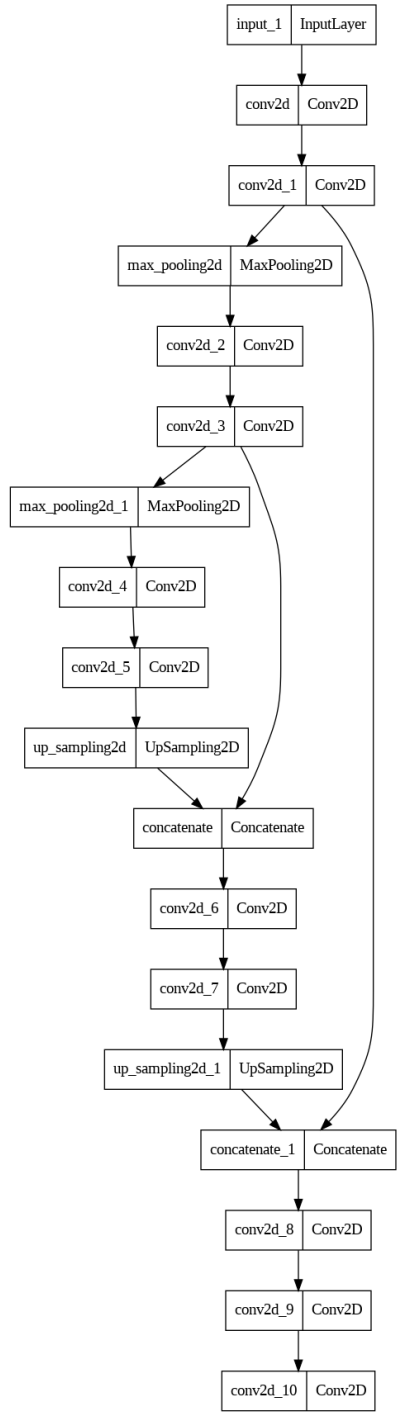
One-hot-encoding to be further used for metrics compatibility.

MODEL FROM SCRATCH



UNET INSPIRED ARCHITECTURE

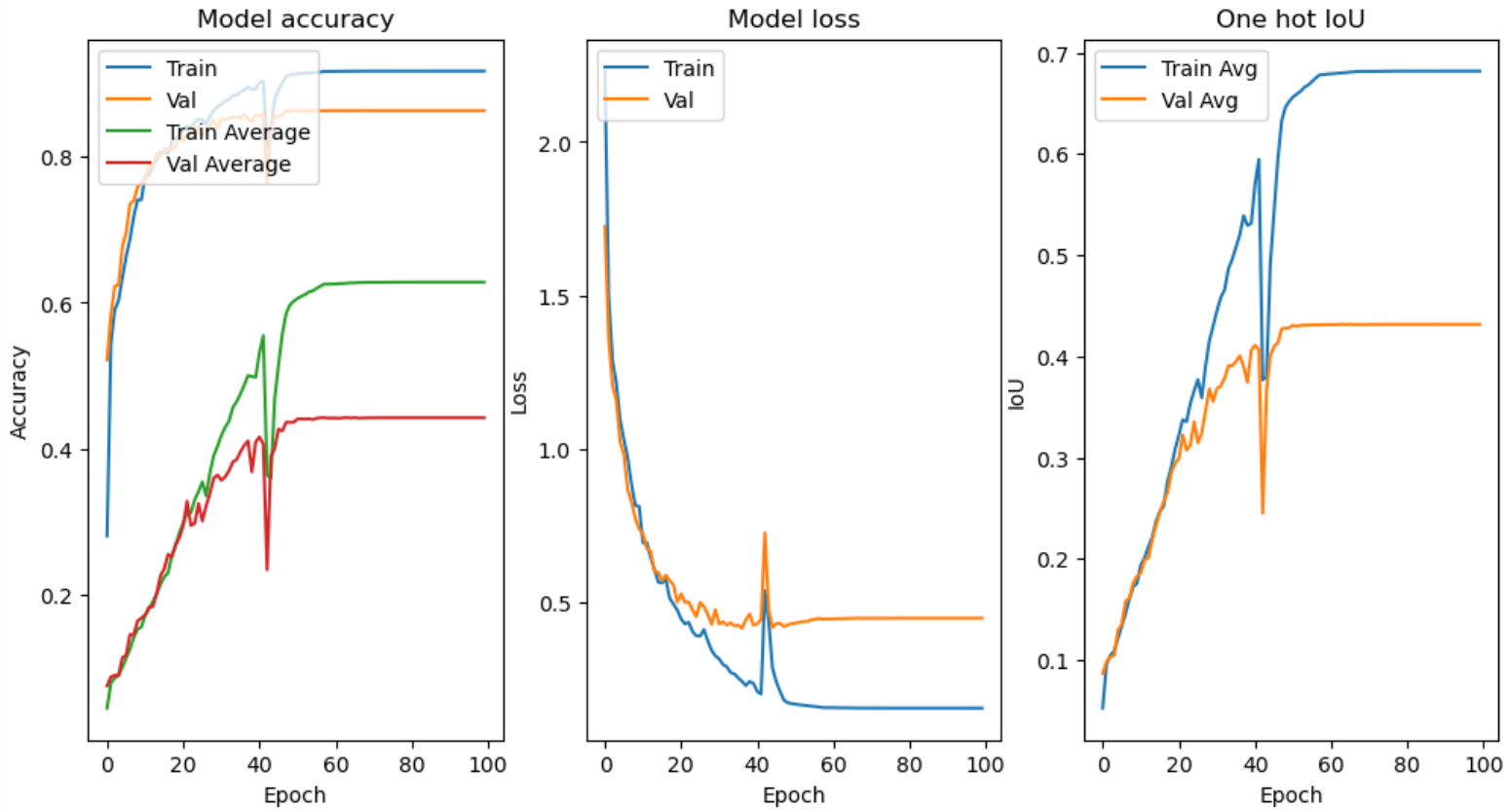




UNET
BASE VERSION

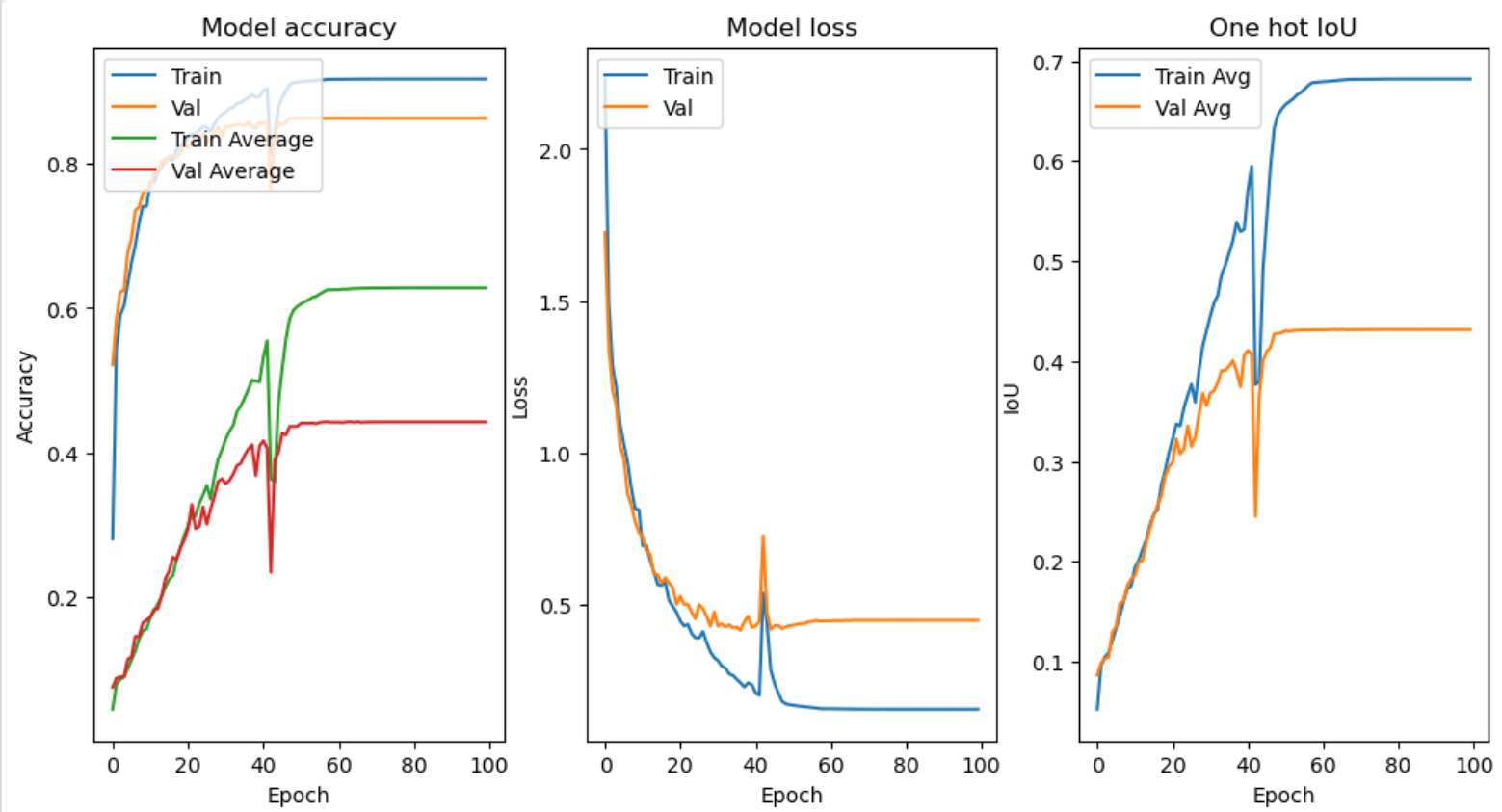
The evaluation of the model is based on:

- **Overall** pixel **accuracy**
- **Mean** pixel **accuracy**
- Mean **IoU**



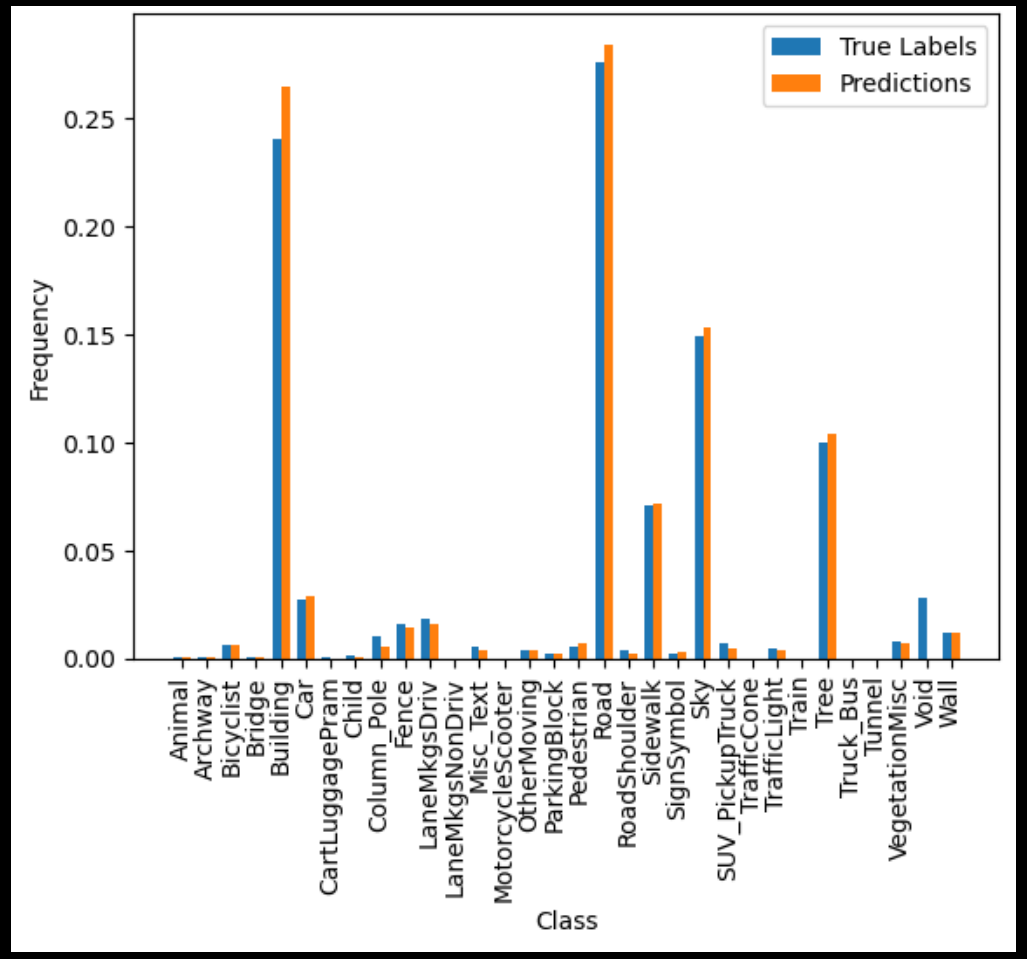
UNET
BASE VERSION

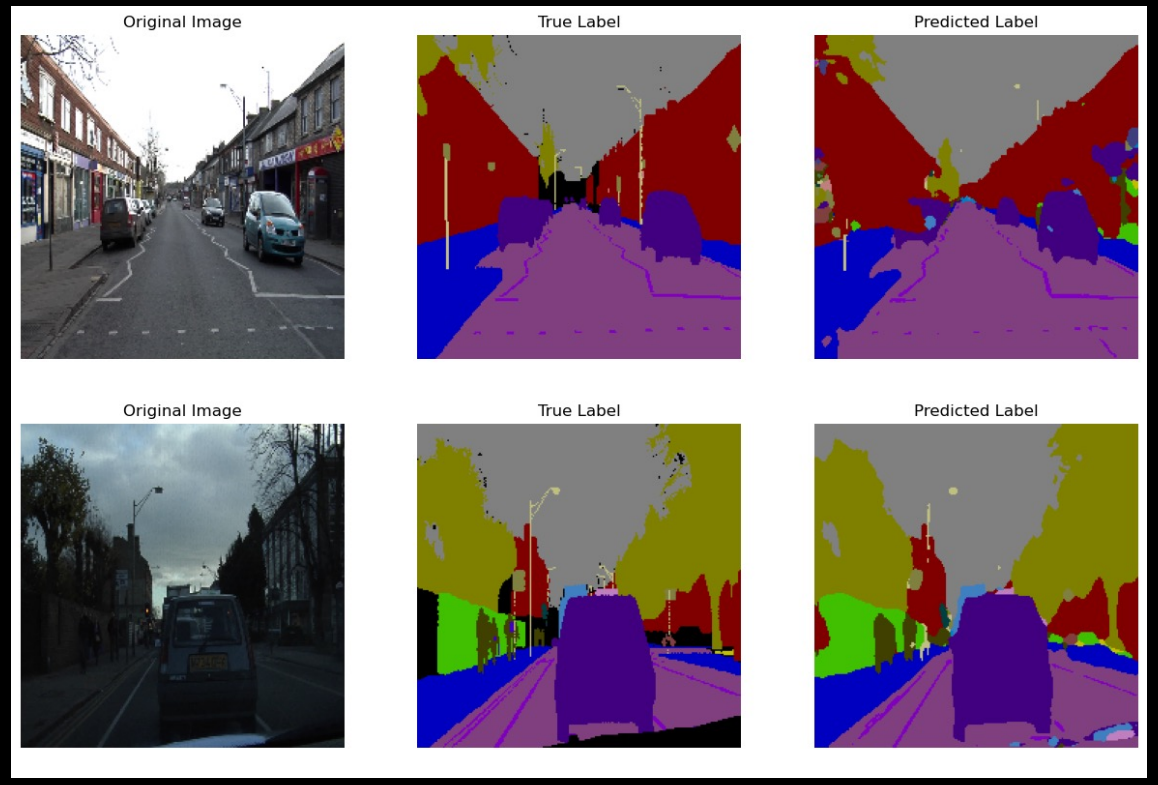
- Batch size = 8
- Epochs = 100
- LR=0.001 (can vary on training)
- Loss: categorical crossentropy (with mask)
- Optimizer: Adam
- Activation: ReLU



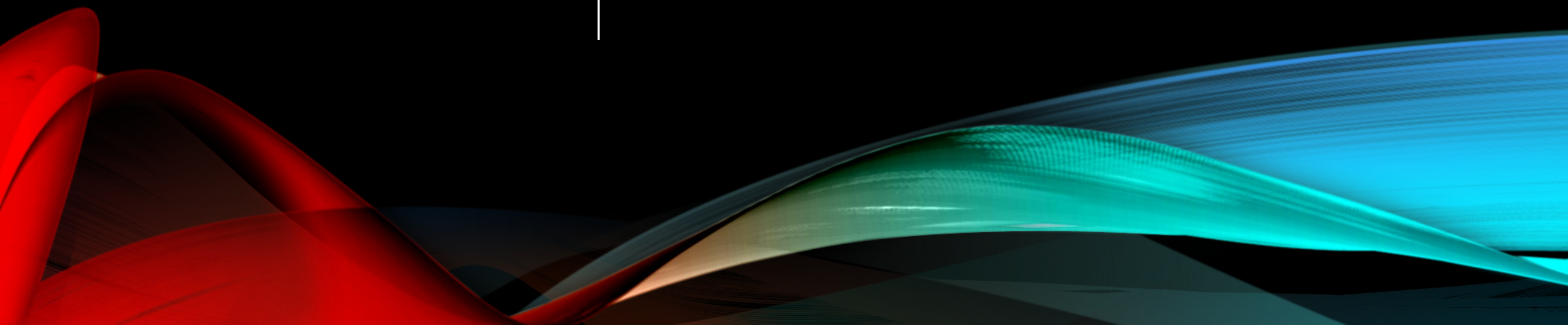
PREDICTIONS VISUALIZATION

Note: Void class is masked





DATA AUGMENTATION



DATA AUGMENTATION

Implementation is performed through a **Custom Class**, in order to augment both raw images and labels.

Applied operations:

1. **Brightness**
2. **Contrast**
3. **Saturation**
4. **Gaussian noise**
5. **Random flip**
6. **Crop (and resize)**
7. **Rotate (and resize)**

Note:

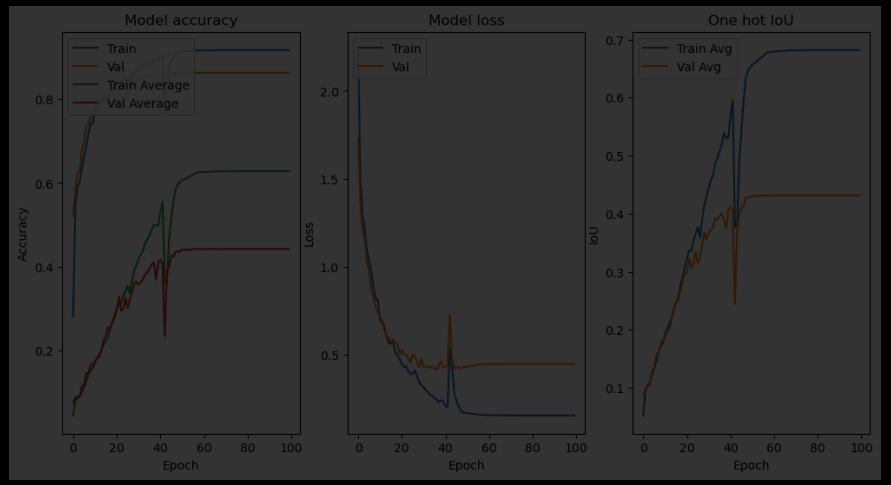
a) Operations in **blue** preserve the label, operations in **red** do not.

b) Cropping and rotation are applied with **different interpolations**

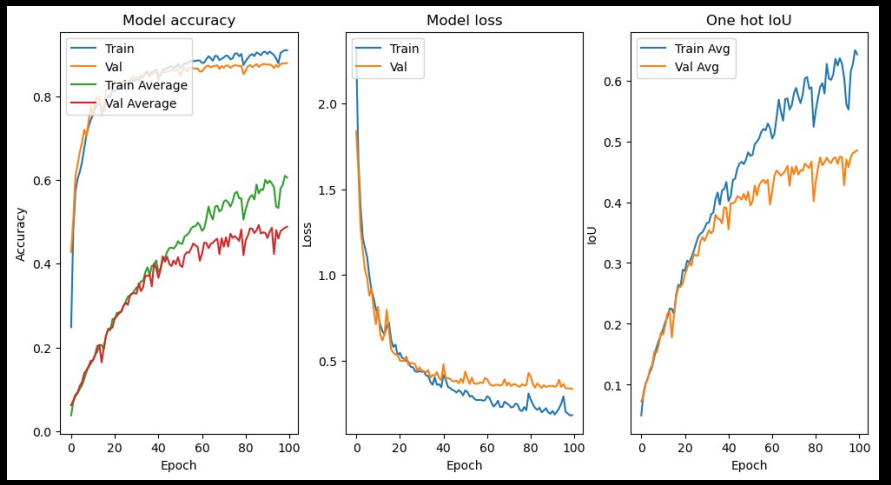
c) Resizing performed to avoid handling **black edges**

DATA AUGMENTATION

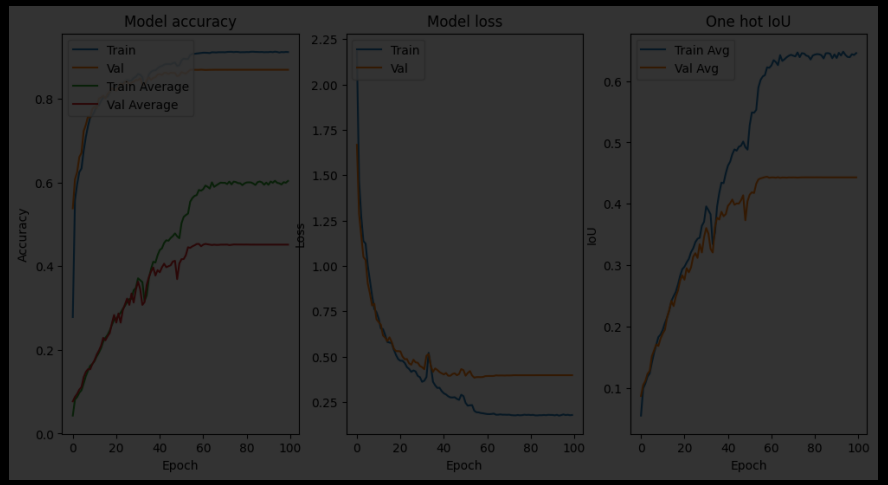
Base Version



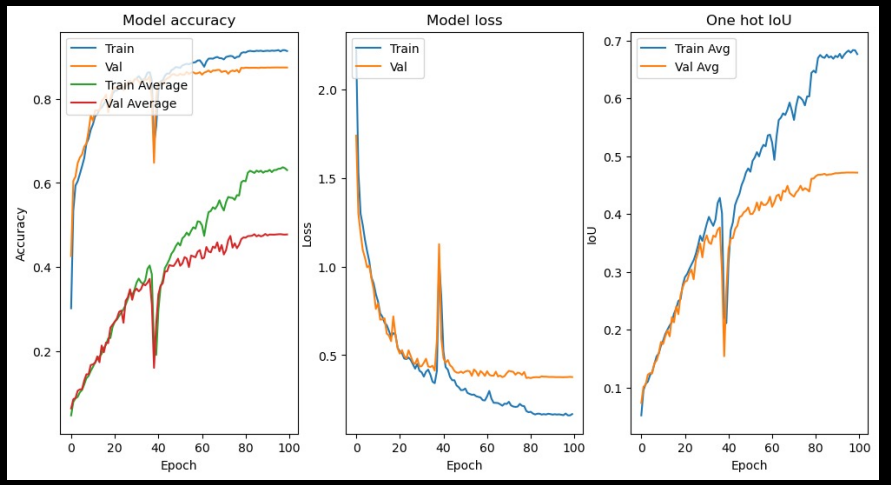
Crop



Horizontal flip

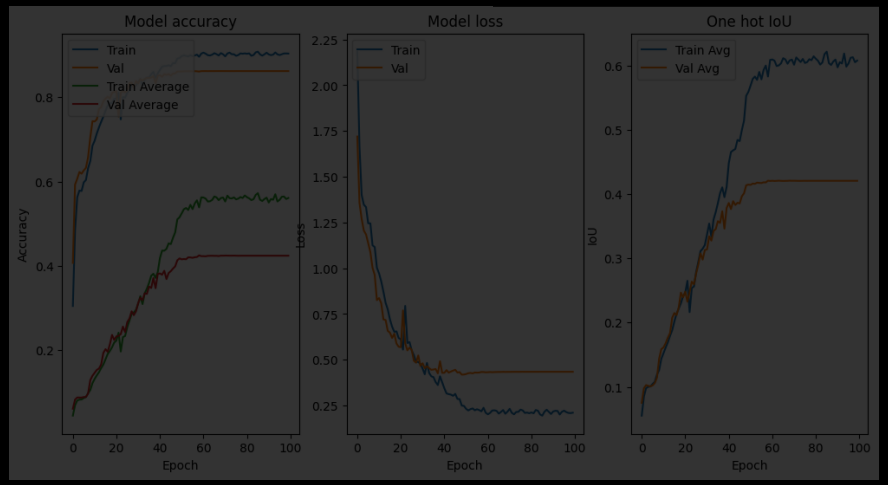


Rotate

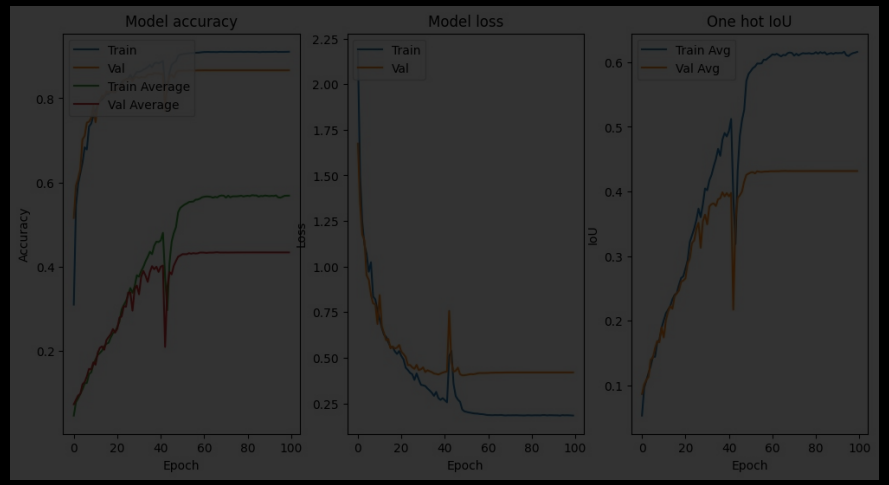


DATA AUGMENTATION

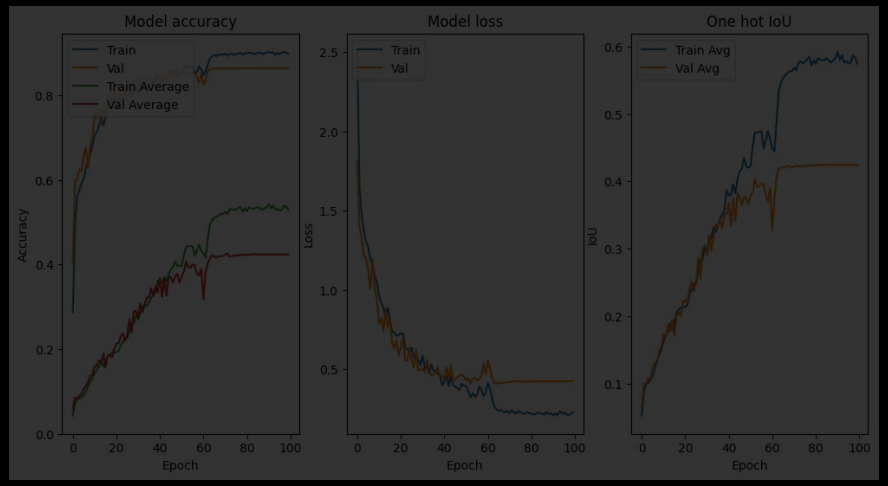
Brightness



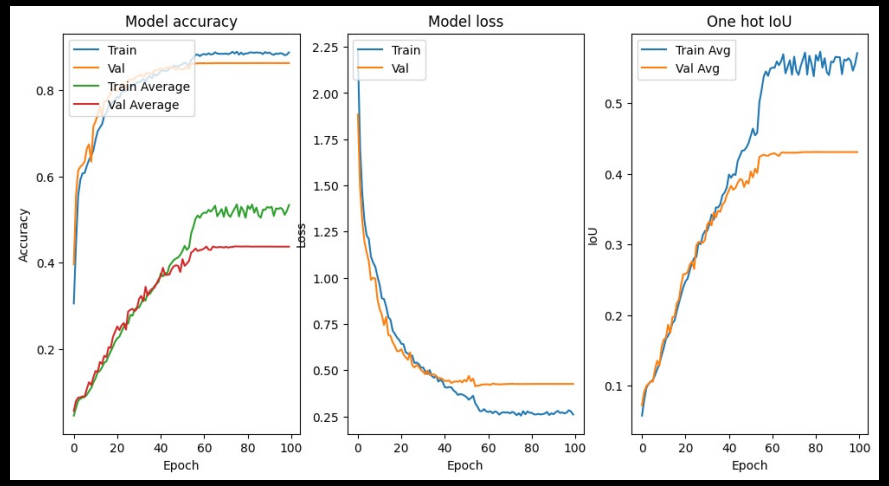
Saturation



Contrast

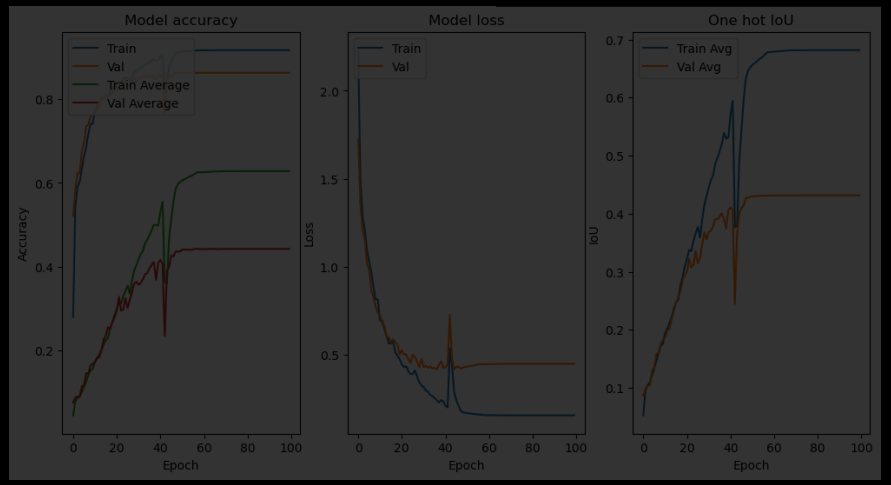


Gaussian Noise

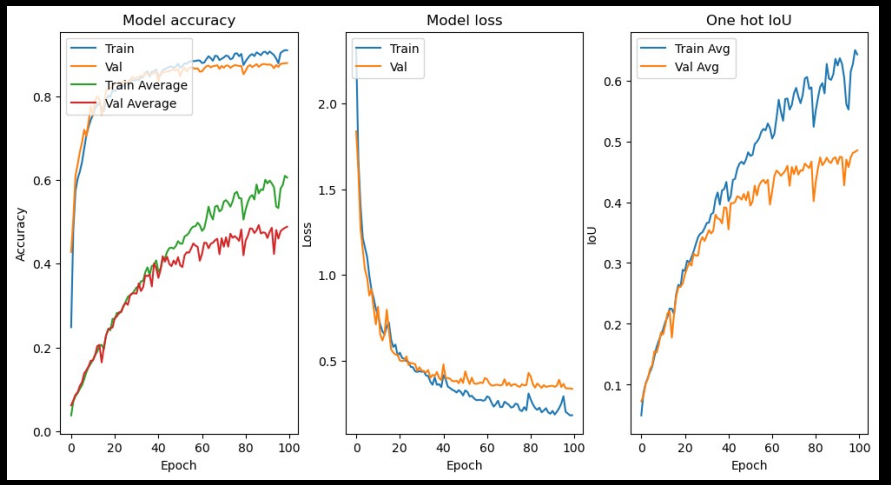


DATA AUGMENTATION

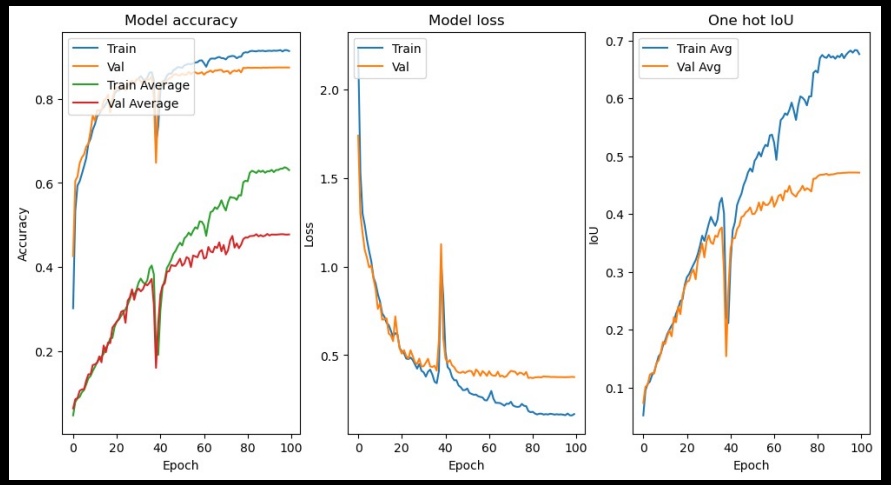
Base Version



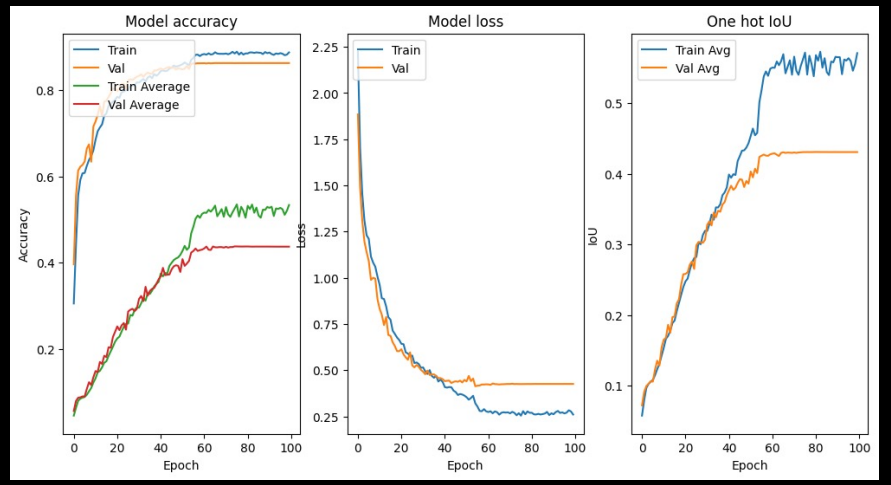
Crop



Rotate



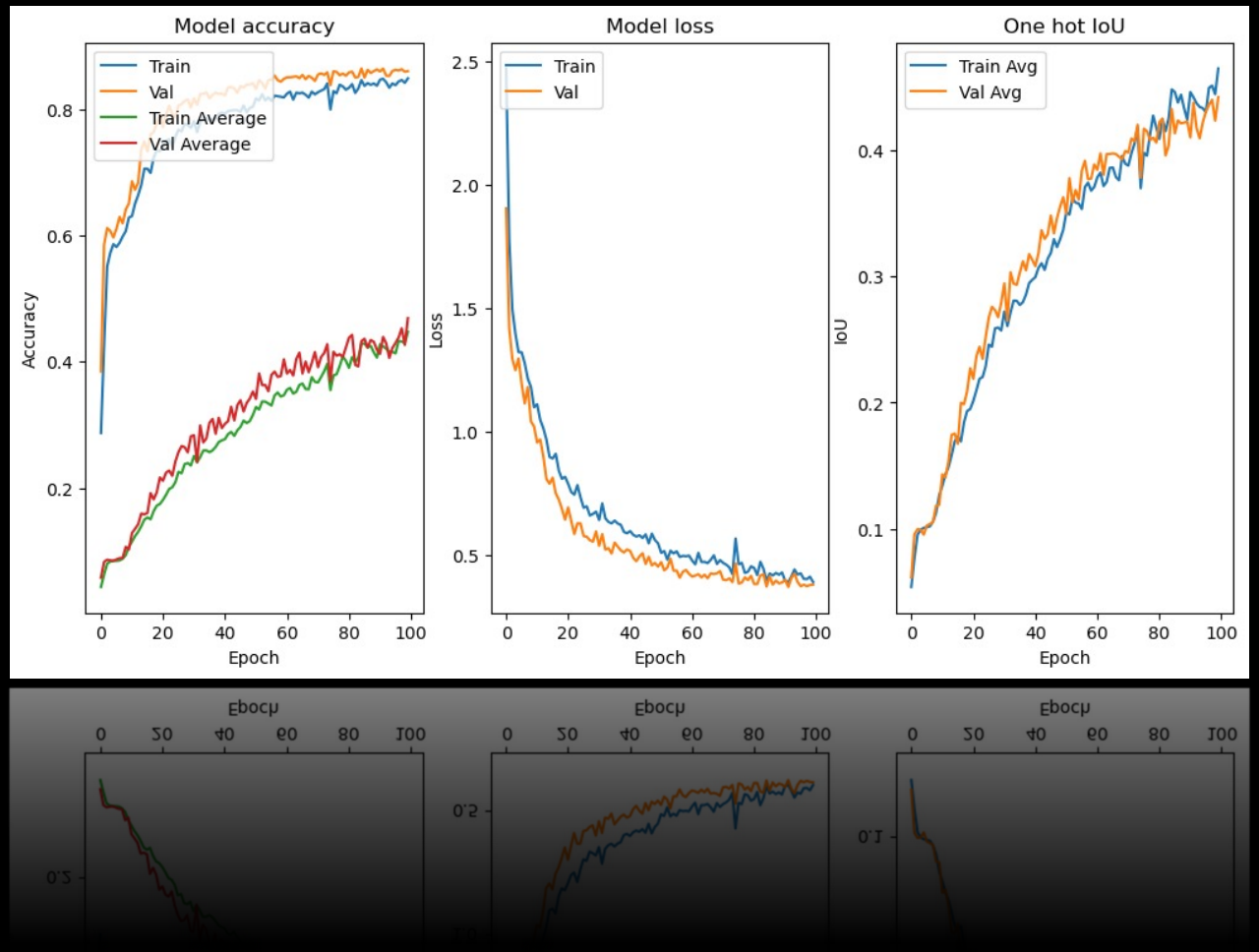
Gaussian Noise



DATA AUGMENTATION

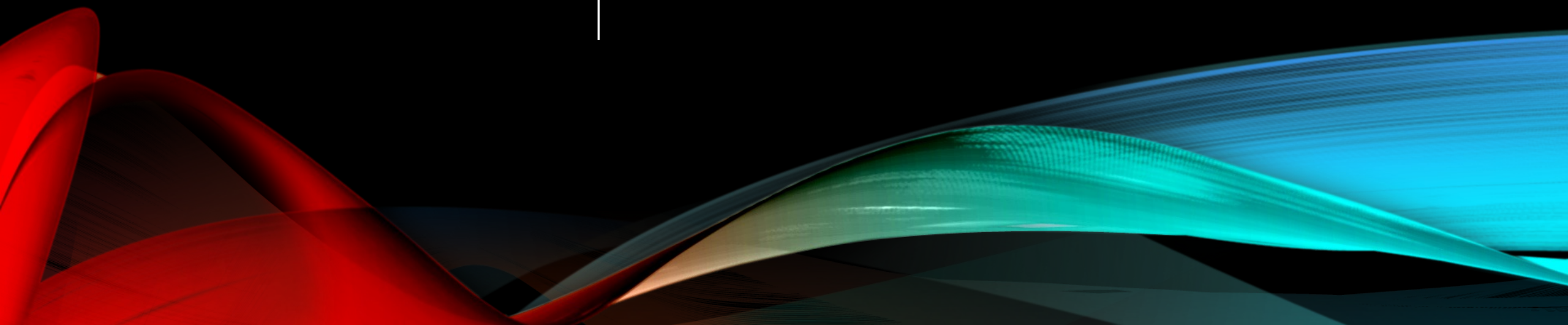
Final result with co-application of the three best DAs

We apply the best 3 performing DAs together: **CROP**, **ROTATION**, **GAUSSIAN NOISE**.



The **overfitting** phenomenon is significantly **reduced**.

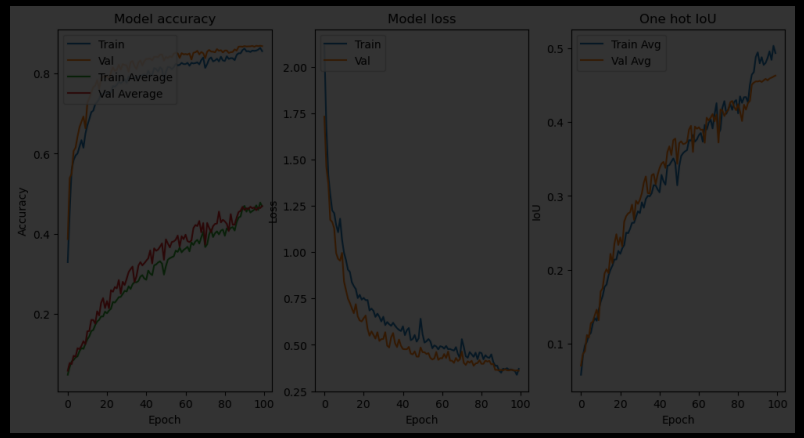
HYPERPARAMETERS OPTIMIZATION



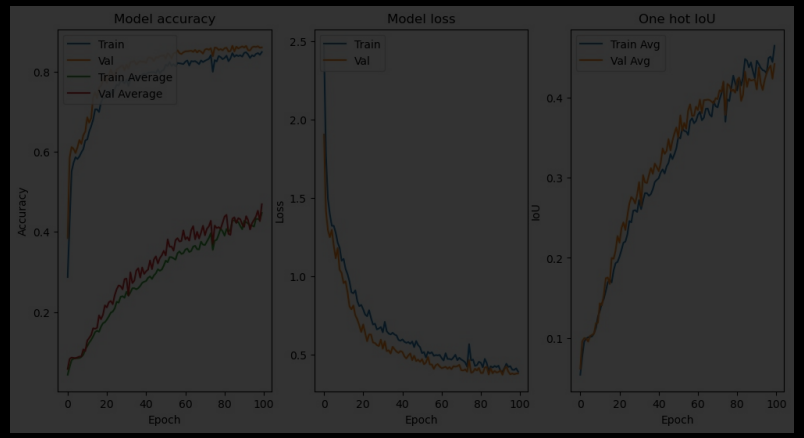
BATCH SIZES

Different values of the **batch size** are implemented

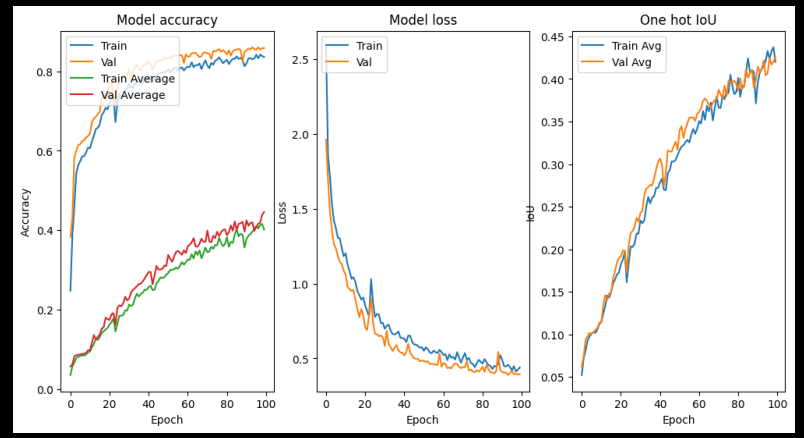
Batch size=**4**



Batch size=**8**

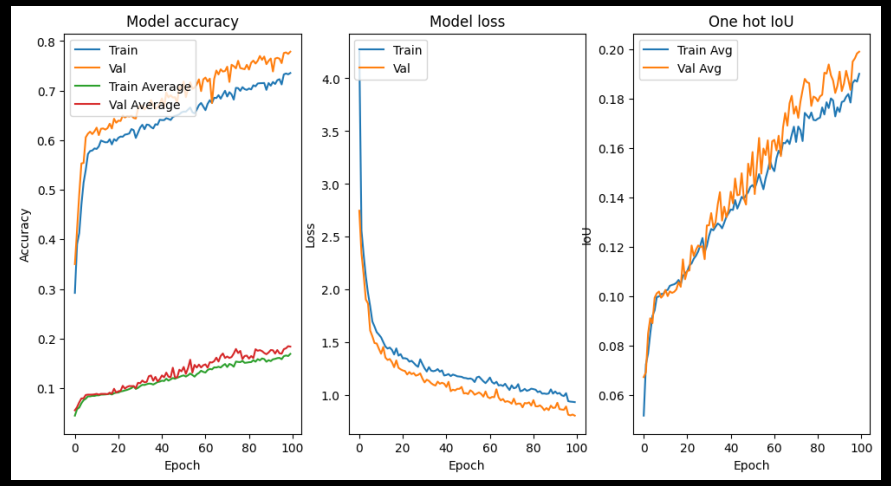


Batch size=**16**

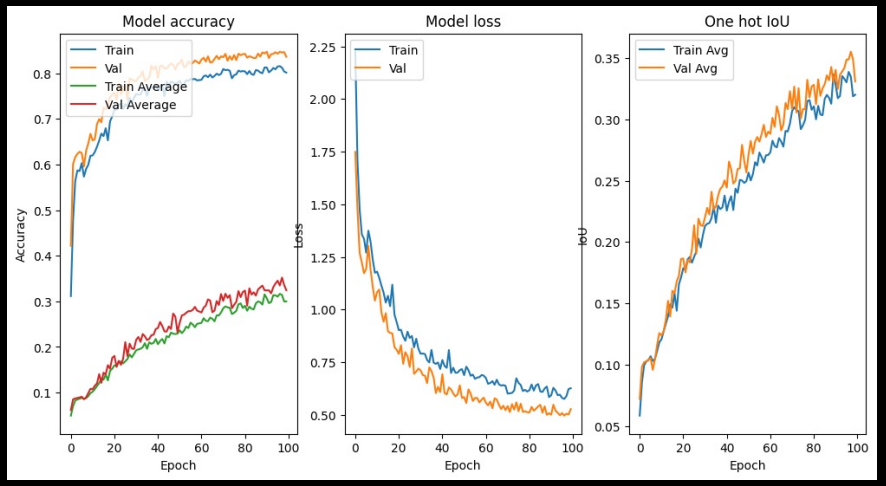


REGULARIZATION

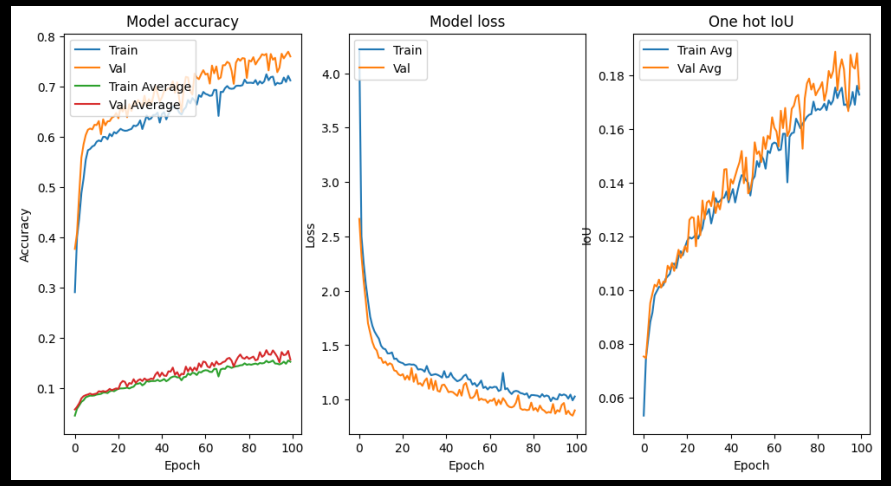
L1 (Lasso Regression)



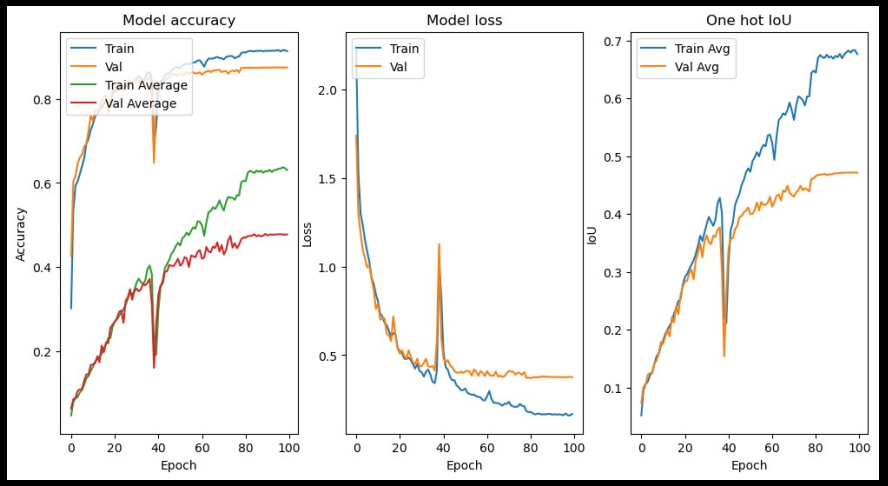
L2 (Ridge Regression)



L1L2 (both penalties)



Dropout

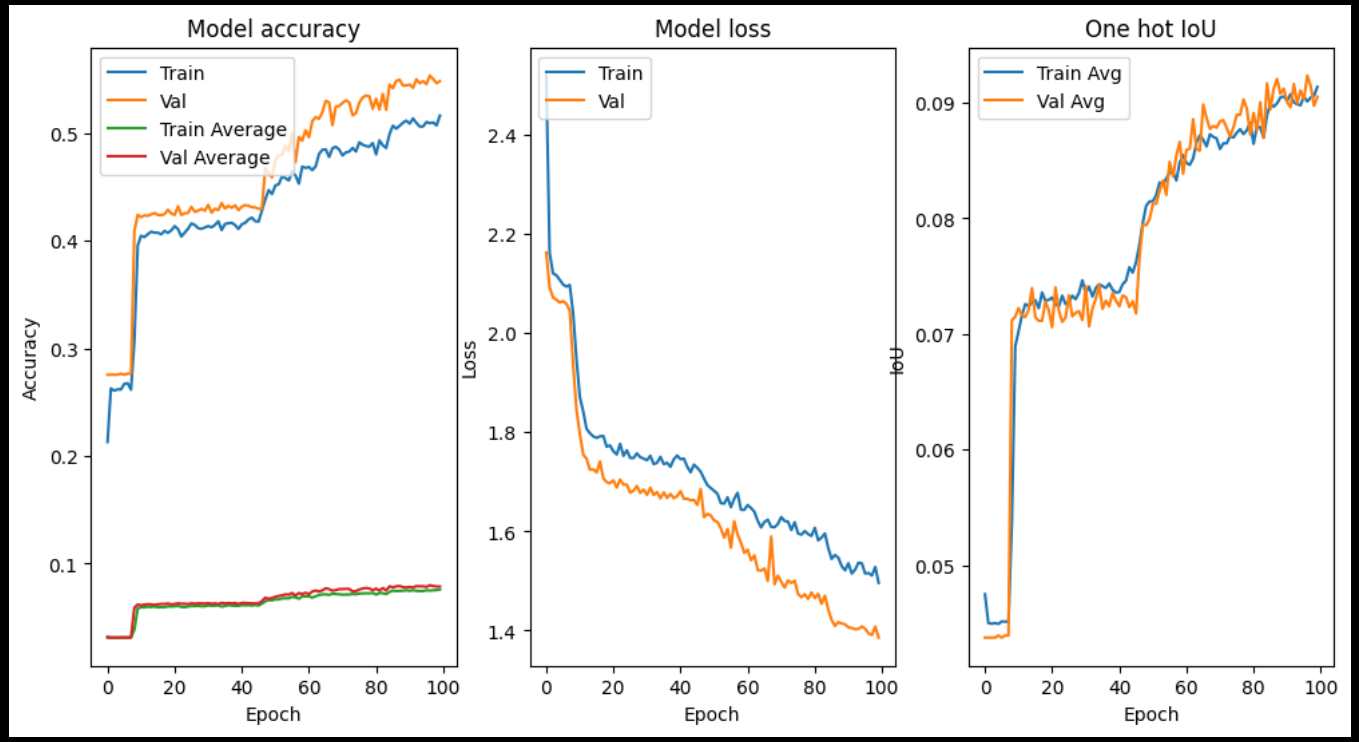


ACTIVATION FUNCTIONS

Sigmoid/Tgh activation function experimented as an alternative

Dropped due to poor performances

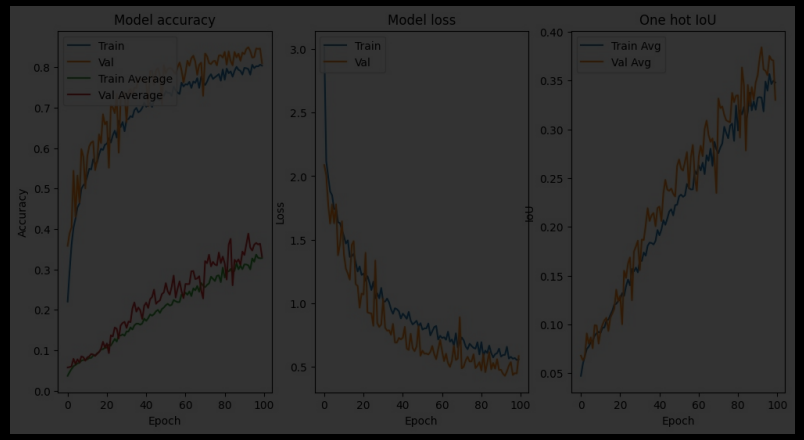
Future perspectives: **trainable** activation functions



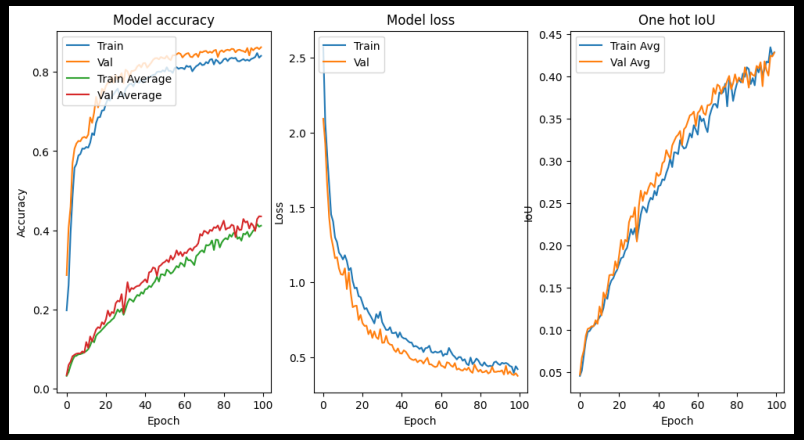
OPTIMIZERS

Different **optimizers** are experimented

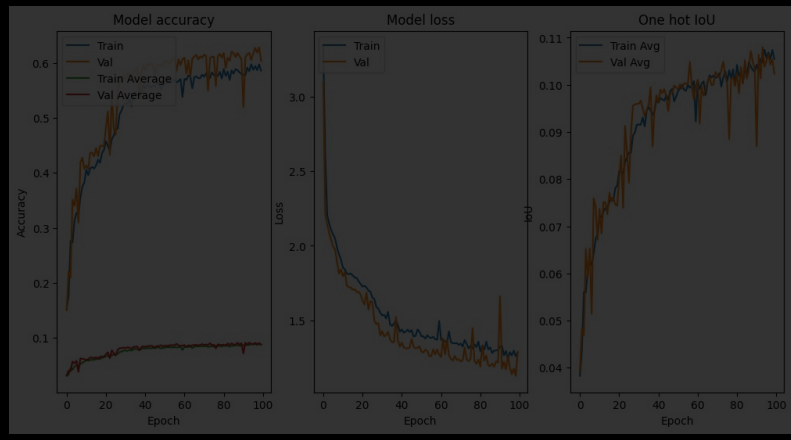
RMSprop



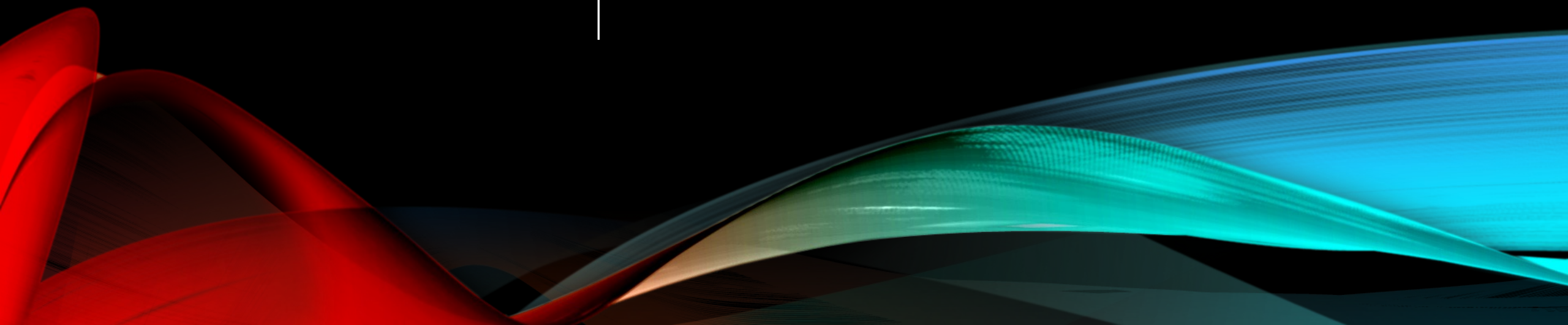
Adam



SGD



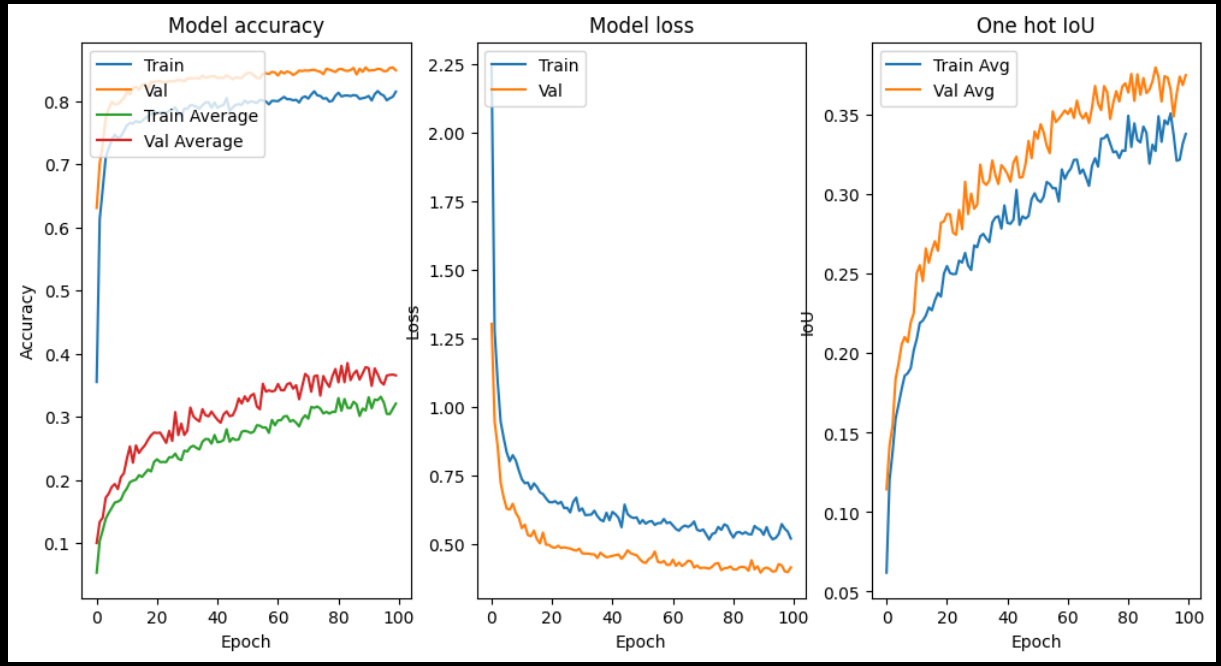
PRETRAINED MODELS



MOBILENETV2

Implemented through a **MobileNetV2** pre-trained on ImageNet for the encoding path.

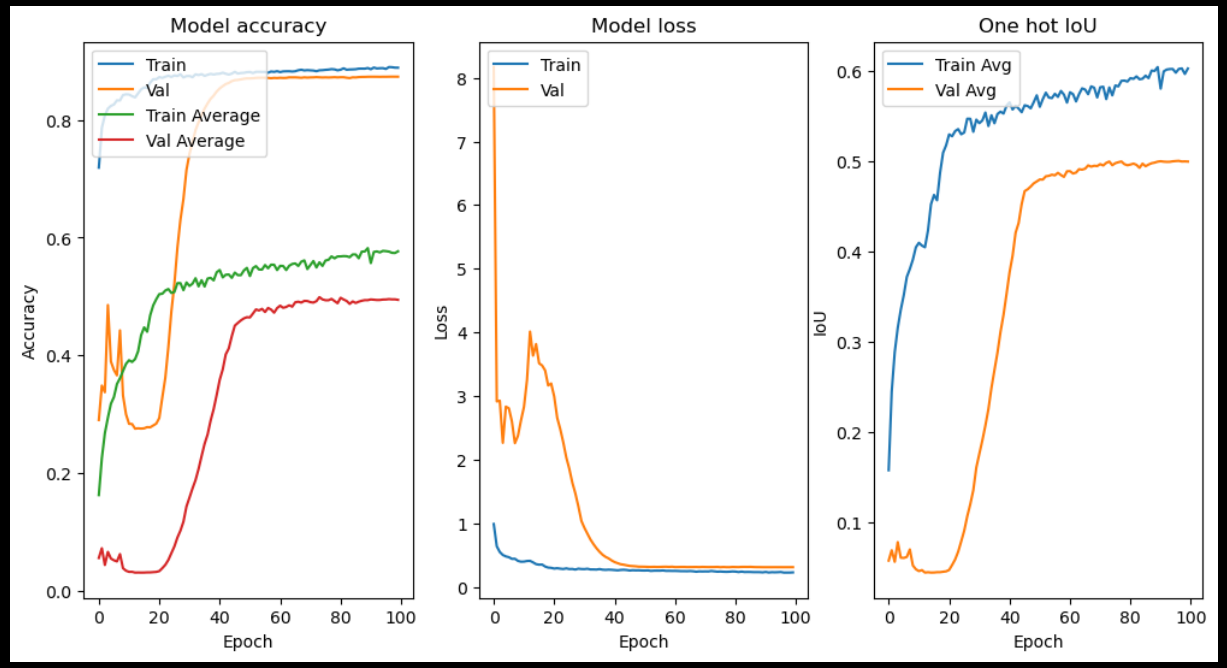
```python
def create_model(input_shape, num_classes):
    base_model = MobileNetV2(include_top=False, input_shape=input_shape, weights='imagenet')

    for layer in base_model.layers:
        layer.trainable = False

    layer_dict = dict([(layer.name, layer) for layer in base_model.layers])

    block_names = ['block_1_expand_relu', 'block_3_expand_relu',
                   'block_6_expand_relu', 'block_13_expand_relu', 'block_16_project']
    blocks = [layer_dict[name].output for name in block_names]

    x = blocks[-1]

    for i, block in enumerate(reversed(blocks[:-1])):
        x = Conv2DTranspose(256 // (2 ** i), (3, 3), strides=2, padding='same')(x)
        x = concatenate([x, block])

    x = Conv2DTranspose(num_classes, (3, 3), strides=2, padding='same', activation='softmax')(x)

    model = Model(inputs=base_model.input, outputs=x)
    return model

model = create_model((224, 224, 3), 32)
model.summary()
```



DEEPLAB

Implemented through a **ResNet** pre-trained on ImageNet for the encoding path and **ASPP** (Atrous Spatial Pyramid Pooling)

```python
def ASPP(inputs, filters=256):
    """ Atrous Spatial Pyramid Pooling """

    y1 = Conv2D(filters, 1, padding="same", use_bias=False)(inputs)
    y1 = BatchNormalization()(y1)
    y1 = Activation("relu")(y1)

    y2 = Conv2D(filters, 3, padding="same", use_bias=False, dilation_rate=6)(inputs)
    y2 = BatchNormalization()(y2)
    y2 = Activation("relu")(y2)

    y3 = Conv2D(filters, 3, padding="same", use_bias=False, dilation_rate=12)(inputs)
    y3 = BatchNormalization()(y3)
    y3 = Activation("relu")(y3)

    y4 = Conv2D(filters, 3, padding="same", use_bias=False, dilation_rate=18)(inputs)
    y4 = BatchNormalization()(y4)
    y4 = Activation("relu")(y4)

    y5 = Conv2D(filters, 1, padding="same", use_bias=False)(inputs)
    y5 = BatchNormalization()(y5)
    y5 = Activation("relu")(y5)

    y = Concatenate()([y1, y2, y3, y4, y5])

    y = Conv2D(filters, 1, padding="same", use_bias=False)(y)
    y = BatchNormalization()(y)
    y = Activation("relu")(y)

    return y

def DeepLab(input_shape, num_classes):
    inputs = Input(shape=input_shape)

    base_model = ResNet50(weights='imagenet', include_top=False, input_tensor=inputs)

    x = base_model.get_layer('conv4_block6_out'). output
    x = ASPP(x)
    x = UpSampling2D(size=(4, 4), interpolation='bilinear')(x)
    x = UpSampling2D(size=(4, 4), interpolation='bilinear')(x)

    x = Conv2D(num_classes, 1, padding="same")(x)
    x = Activation('softmax')(x)

    model = Model(inputs=inputs, outputs=x)
```



RESULTS

MODEL EVALUATION

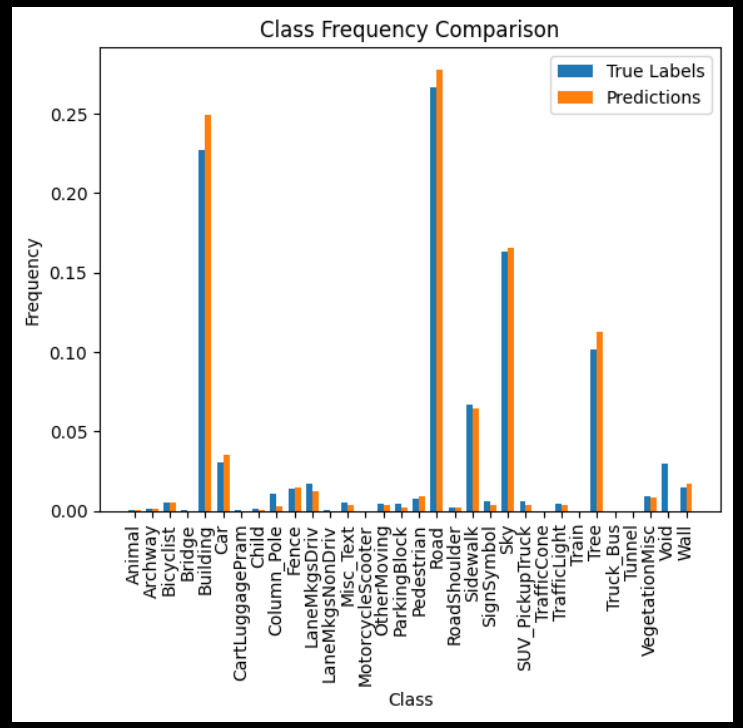
The selected models are evaluated on the **test set** (composed of 150 unseen images).

The files **.keras** are read with the weights resulting from model training in the evaluation notebook

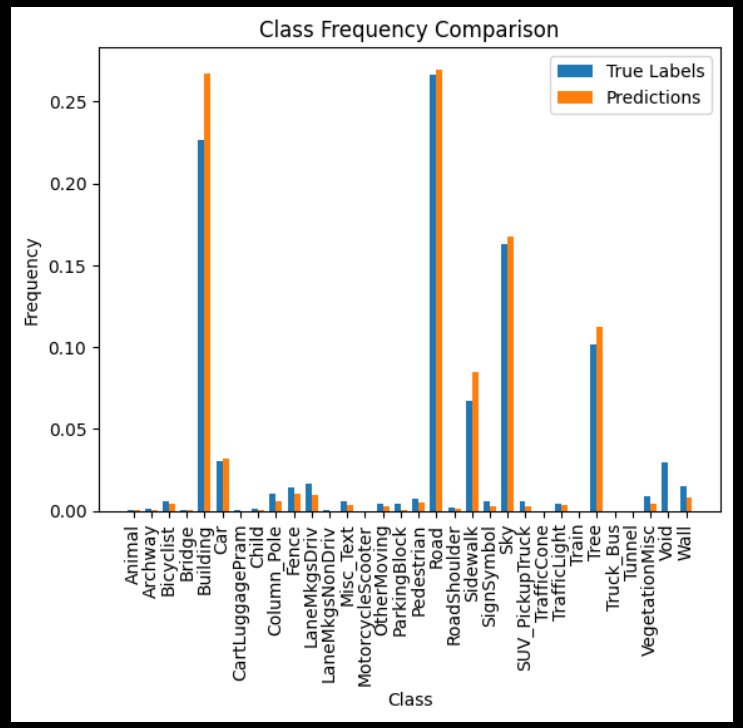
The **decorator** register keras serializable (and after deserializes) the custom metrics and the loss function

HISTOGRAMS

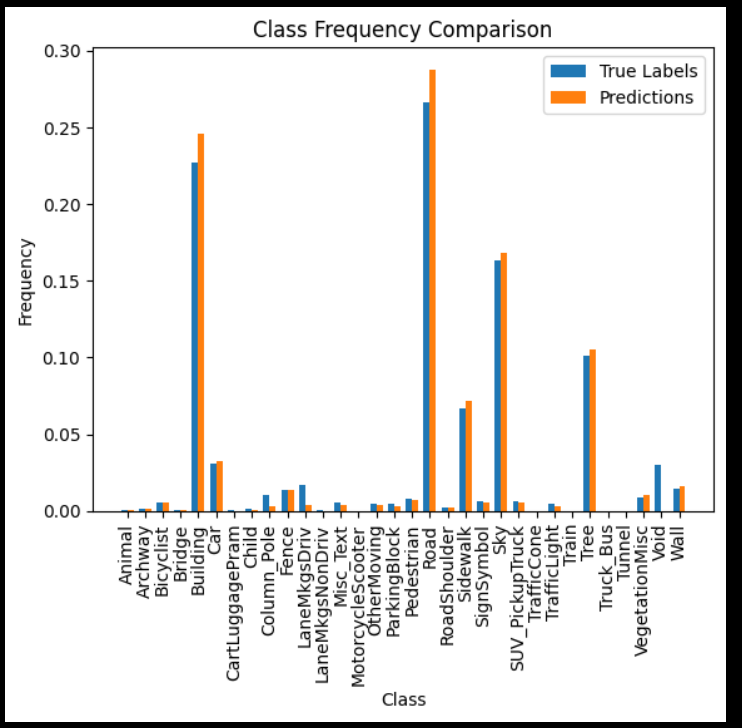
UNet



MobileNetV2



DeepLab



CLASSIFICATION REPORTS

UNet

	Class	Precision	Recall	F1-score	Support	IoU
0	Animal	0.63027	0.454227	0.52796	2567	0.358659
1	Archway	0.505793	0.430025	0.464842	9339	0.302797
2	Bicyclist	0.715395	0.71504	0.715218	42399	0.556684
3	Bridge	0.864286	0.322452	0.469675	1501	0.306912
4	Building	0.849673	0.933506	0.889619	1695047	0.801183
5	Car	0.755548	0.872903	0.809997	227638	0.680668
6	CartLuggagePram	0.0	0.0	0.0	2488	0.0
7	Child	0.77643	0.262511	0.392363	9512	0.244062
8	Column_Pole	0.528857	0.16027	0.245992	79129	0.140246
9	Fence	0.593746	0.61614	0.604735	104835	0.43342
10	LaneMkgsDriv	0.786957	0.597665	0.679372	125644	0.514431
11	LaneMkgsNonDriv	0.0	0.0	0.0	1718	0.0
12	Misc_Text	0.561038	0.369862	0.44582	42359	0.286852
13	MotorcycleScooter	0.0	0.0	0.0	771	0.0
14	OtherMoving	0.571131	0.467193	0.51396	32310	0.345859
15	ParkingBlock	0.658935	0.365442	0.470145	33428	0.307313
16	Pedestrian	0.432646	0.50923	0.467825	57041	0.305334
17	Road	0.928516	0.968874	0.948266	1989511	0.901622
18	RoadShoulder	0.698743	0.579328	0.633457	17560	0.463547
19	Sidewalk	0.894655	0.863661	0.878885	501588	0.783938
20	SignSymbol	0.770984	0.517598	0.619379	45288	0.448623
21	Sky	0.947798	0.960015	0.953868	1218670	0.911804
22	SUVPickupTruck	0.654837	0.428938	0.518344	44482	0.349841
23	TrafficCone	0.0	0.0	0.0	497	0.0
24	TrafficLight	0.739056	0.652044	0.692829	32234	0.530021
25	Train	0.0	0.0	0.0	0	NaN
26	Tree	0.786604	0.874337	0.828153	757581	0.706708
27	Truck_Bus	0.0	0.0	0.0	0	NaN
28	Tunnel	0.0	0.0	0.0	0	NaN
29	VegetationMisc	0.58453	0.537709	0.560143	66629	0.389026
30	Void	0.0	0.0	0.0	224274	0.0
31	Wall	0.635803	0.728717	0.679097	110184	0.514115
32						
33	Accuracy	0.85964	0.85964	0.85964	7476224	NaN
34	Macro Avg	0.527257	0.443365	0.469061	7476224	NaN
35	Weighted Avg	0.830048	0.85964	0.841719	7476224	NaN

MobileNetV2

	Class	Precision	Recall	F1-score	Support	IoU
0	Animal	0.514068	0.263342	0.348274	2567	0.210855
1	Archway	0.518486	0.375415	0.435501	9339	0.278364
2	Bicyclist	0.715744	0.547371	0.620336	42399	0.449628
3	Bridge	0.285933	0.373751	0.323997	1501	0.193315
4	Building	0.801557	0.944374	0.867125	1695047	0.765419
5	Car	0.778771	0.808367	0.793293	227638	0.657403
6	CartLuggagePram	0.123457	0.004019	0.007785	2488	0.003908
7	Child	0.60629	0.226976	0.330299	9512	0.197819
8	Column_Pole	0.289868	0.162519	0.208269	79129	0.116239
9	Fence	0.694307	0.521954	0.595918	104835	0.424418
10	LaneMkgsDriv	0.77768	0.436209	0.558916	125644	0.387844
11	LaneMkgsNonDriv	0.164384	0.034924	0.057609	1718	0.029659
12	Misc_Text	0.471979	0.323284	0.383731	42359	0.237417
13	MotorcycleScooter	0.771739	0.092088	0.164542	771	0.089646
14	OtherMoving	0.607595	0.356546	0.449386	32310	0.289811
15	ParkingBlock	0.497119	0.028389	0.053711	33428	0.027597
16	Pedestrian	0.406249	0.26509	0.320829	57041	0.191064
17	Road	0.92515	0.937395	0.931232	1989511	0.871314
18	RoadShoulder	0.69593	0.316458	0.435075	17560	0.278017
19	Sidewalk	0.728327	0.919019	0.812636	501588	0.684403
20	SignSymbol	0.83162	0.423247	0.560985	45288	0.389839
21	Sky	0.935548	0.962204	0.948689	1218670	0.902387
22	SUVPickupTruck	0.646002	0.273144	0.383947	44482	0.237583
23	TrafficCone	0.7	0.028169	0.054159	497	0.027833
24	TrafficLight	0.578622	0.510858	0.542633	32234	0.372338
25	Train	0.0	0.0	0.0	0	NaN
26	Tree	0.780052	0.863135	0.819493	757581	0.694188
27	Truck_Bus	0.0	0.0	0.0	0	NaN
28	Tunnel	0.0	0.0	0.0	0	NaN
29	VegetationMisc	0.722607	0.374432	0.493268	66629	0.327376
30	Void	0.0	0.0	0.0	224274	0.0
31	Wall	0.713969	0.403607	0.515692	110184	0.34743
32						
33	Accuracy	0.836505	0.836505	0.836505	7476224	NaN
34	Macro Avg	0.540095	0.368009	0.406792	7476224	NaN
35	Weighted Avg	0.804433	0.836505	0.81327	7476224	NaN

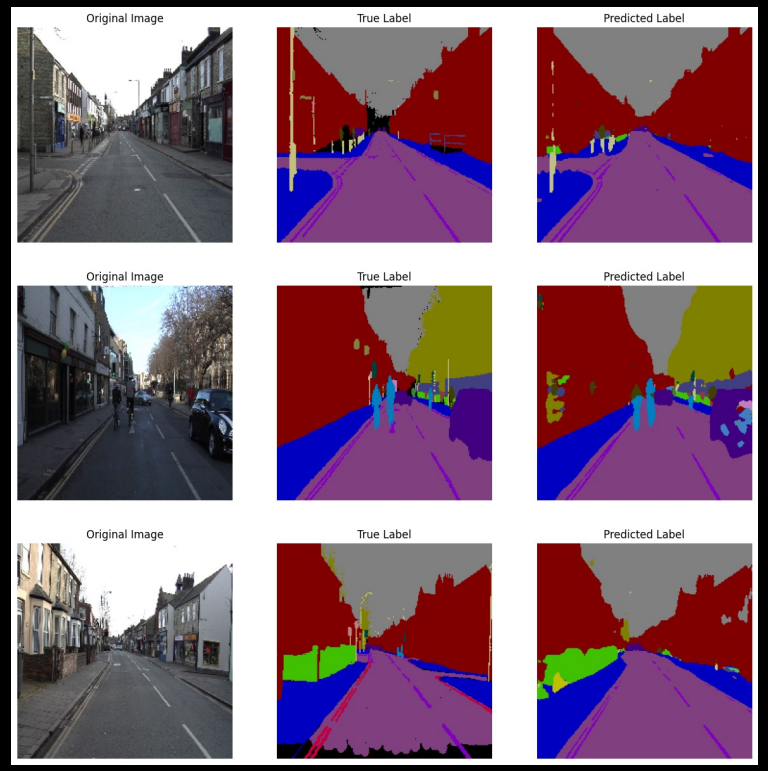
DeepLab

	Class	Precision	Recall	F1-score	Support	IoU
0	Animal	0.475199	0.417998	0.444767	2567	0.285981
1	Archway	0.58459	0.597923	0.591181	9339	0.419629
2	Bicyclist	0.686886	0.705889	0.696258	42399	0.534046
3	Bridge	0.511598	0.793471	0.622095	1501	0.451478
4	Building	0.879791	0.953314	0.915078	1695047	0.843451
5	Car	0.83915	0.903241	0.870017	227638	0.769938
6	CartLuggagePram	0.0	0.0	0.0	2488	0.0
7	Child	0.627219	0.200589	0.303967	9512	0.179222
8	Column_Pole	0.199497	0.054152	0.085182	79129	0.044486
9	Fence	0.80269	0.75946	0.780477	104835	0.639985
10	LaneMkgsDriv	0.604718	0.134459	0.220001	125644	0.123596
11	LaneMkgsNonDriv	0.0	0.0	0.0	1718	0.0
12	Misc_Text	0.56597	0.410137	0.475614	42359	0.312004
13	MotorcycleScooter	0.647793	0.875486	0.744622	771	0.593146
14	OtherMoving	0.617577	0.503714	0.554864	32310	0.383953
15	ParkingBlock	0.778009	0.537005	0.635422	33428	0.465655
16	Pedestrian	0.482171	0.465595	0.473738	57041	0.310391
17	Road	0.901841	0.974891	0.936944	1989511	0.881369
18	RoadShoulder	0.853225	0.683941	0.759262	17560	0.611943
19	Sidewalk	0.87203	0.9313	0.900691	501588	0.819325
20	SignSymbol	0.937369	0.849651	0.891357	45288	0.804008
21	Sky	0.929624	0.957495	0.943354	1218670	0.892782
22	SUVPickupTruck	0.782958	0.721123	0.750769	44482	0.600985
23	TrafficCone	0.0	0.0	0.0	497	0.0
24	TrafficLight	0.681857	0.498542	0.575965	32234	0.40446
25	Train	0.0	0.0	0.0	0	NaN
26	Tree	0.851532	0.882752	0.866861	757581	0.765009
27	Truck_Bus	0.0	0.0	0.0	0	NaN
28	Tunnel	0.0	0.0	0.0	0	NaN
29	VegetationMisc	0.656523	0.784538	0.714844	66629	0.556232
30	Void	0.0	0.0	0.0	224274	0.0
31	Wall	0.711645	0.792547	0.749921	110184	0.599898
32						
33	Accuracy	0.872283	0.872283	0.872283	7476224	NaN
34	Macro Avg	0.546296	0.512163	0.515727	7476224	NaN
35	Weighted Avg	0.835278	0.872283	0.849411	7476224	NaN

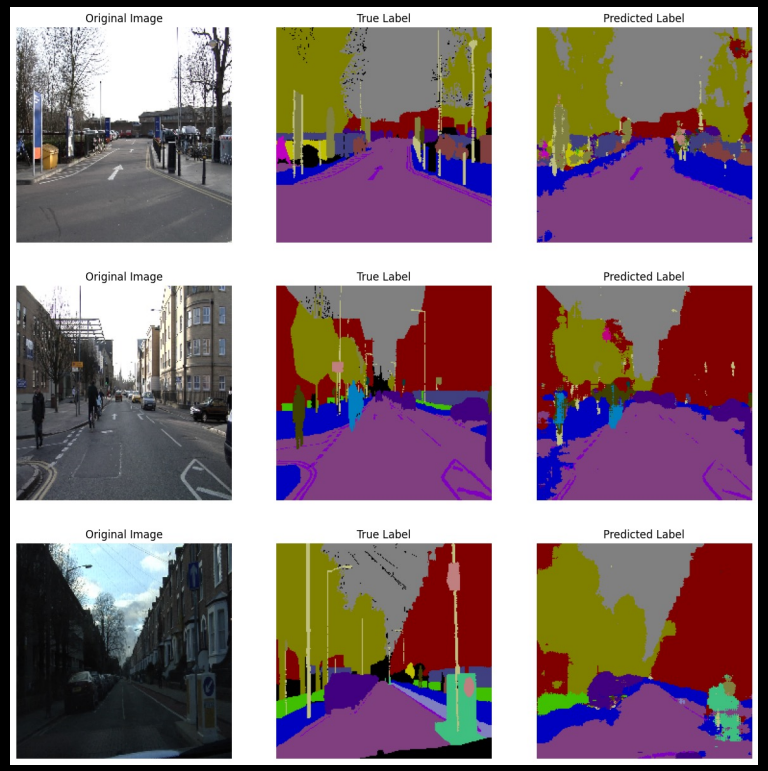
MODELS PREDICTIONS

Samples **visualization**

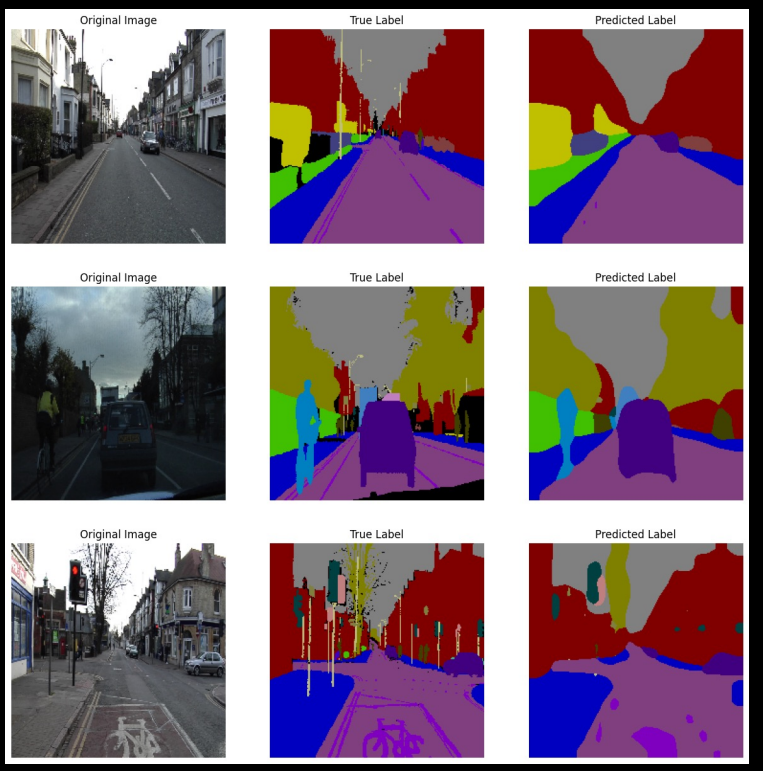
UNet



MobileNetV2

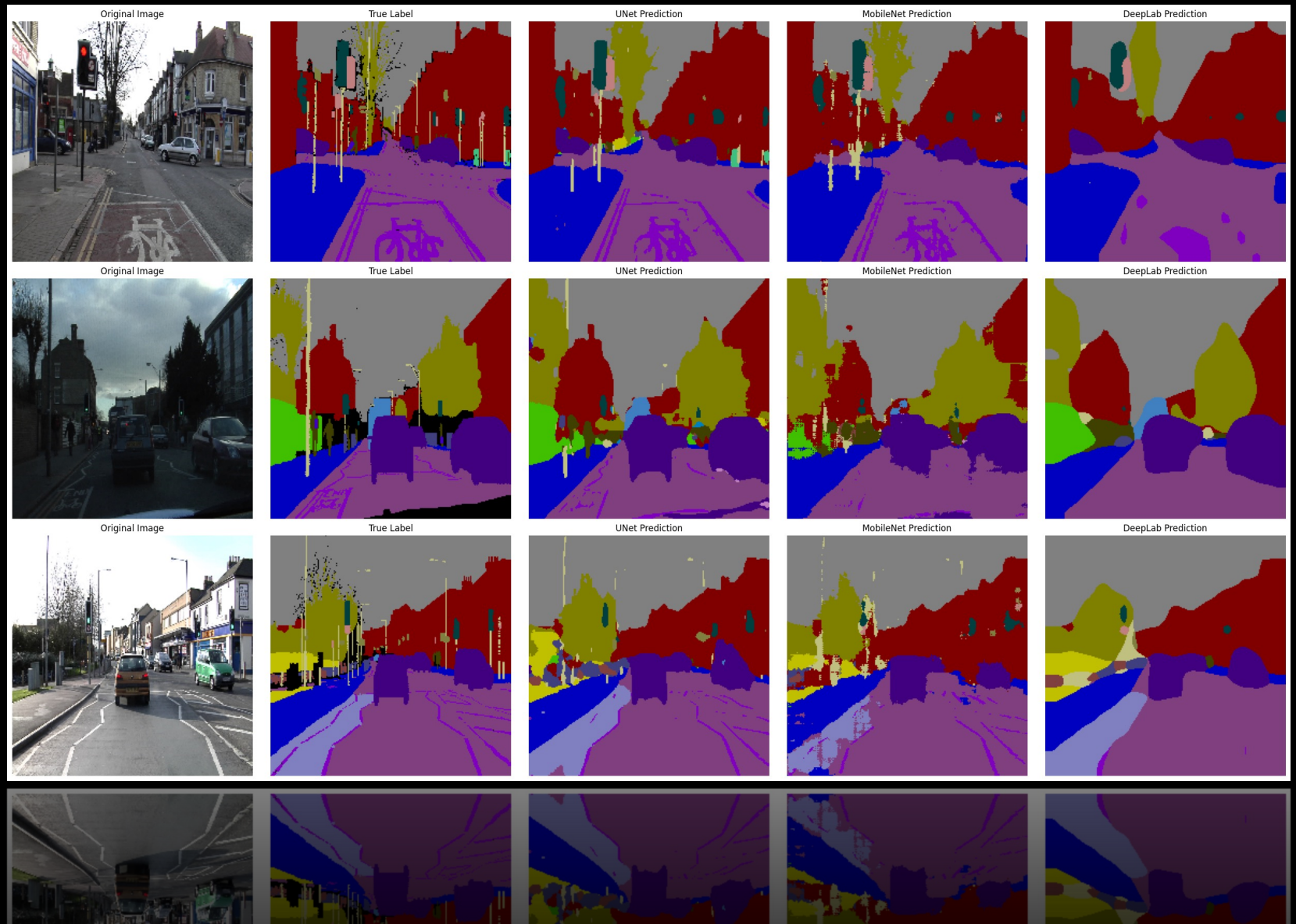


DeepLab

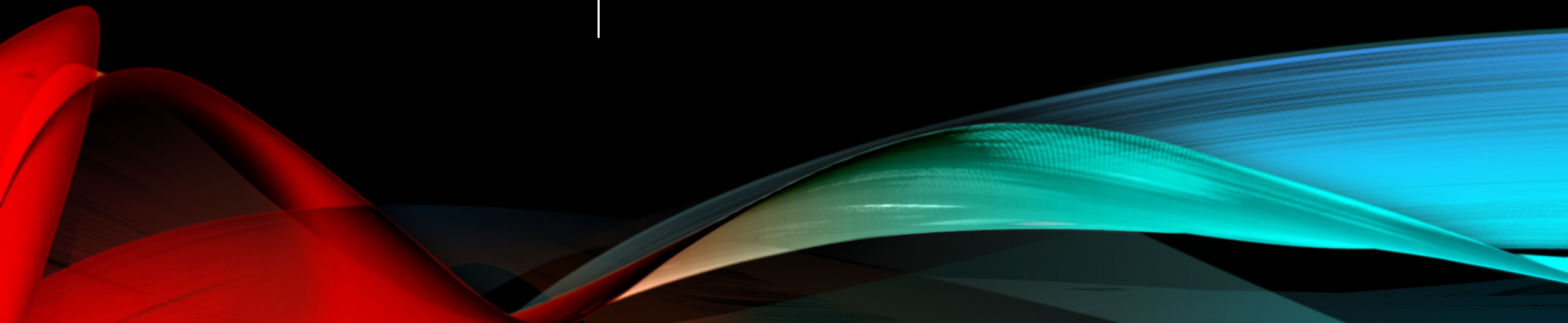


MODELS PREDICTIONS

Overall **comparison**



CONCLUSIONS AND FUTURE PERSPECTIVES



CONCLUSIONS

DeepLab (mean IoU= 0.5242) performs in average **better** than Unet (0.4507) and MobileNetV2 (0.3836)

Hard for the models to distinguish between similar and **rare classes**.

The **easiest** objects to classify are uniform, large ones, characterized by **high contrast.**

CONCLUSIONS

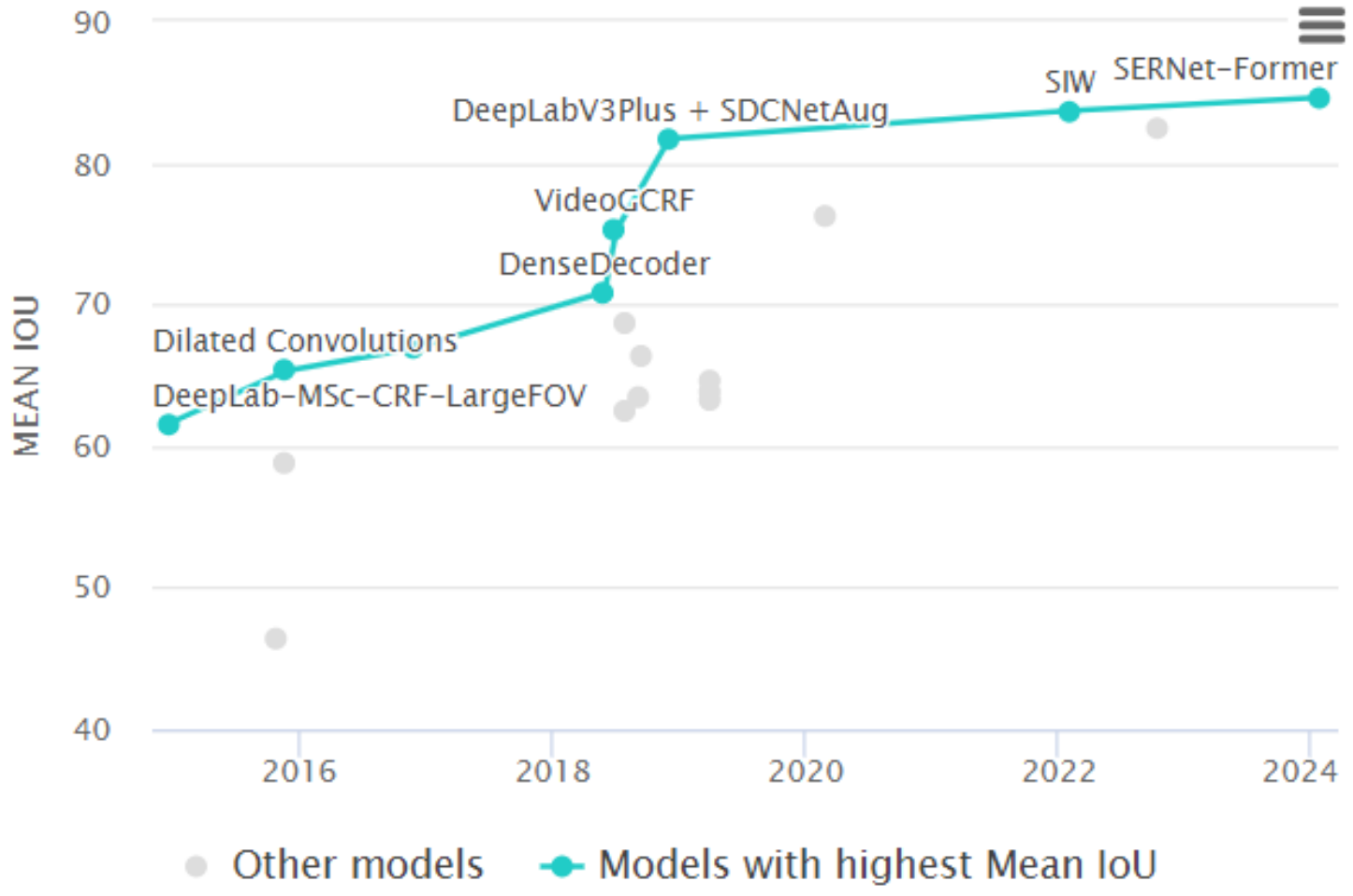
Powerful feature extractor of **DeepLab** allows the minimal confusion between different object classes, w.r.t. **Unet** and **MobileNetV2**.

All models struggle with **shadowed areas** and regions with **complex backgrounds**.

MobileNetV2 is optimized for **low computational** tasks, **pretrained** on structurally different natural images and **not tuned** as Unet is.

FUTURE PERSPECTIVES

- Trainable activation functions
- Combinations of data augmentations and ClassMix
- Hyperparameter optimization via CV or Bayesian optimization
- In-depth analysis of model inference on Void class
- Implementation of other models



THANK YOU FOR YOUR ATTENTION