Automatic Segmentation of Left Ventricle in 2D Echocardiogram Using Deep Learning

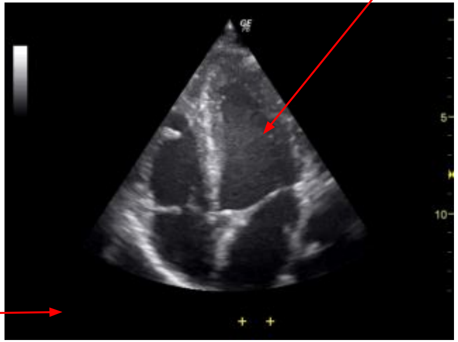
Le Ngoc Tuan Khang

Supervisor: Dr. Tran Quoc Long
Co-Supervisor: Assoc. Prof. Dr. Le Sy Vinh

Introduction: Left Ventricle Segmentation

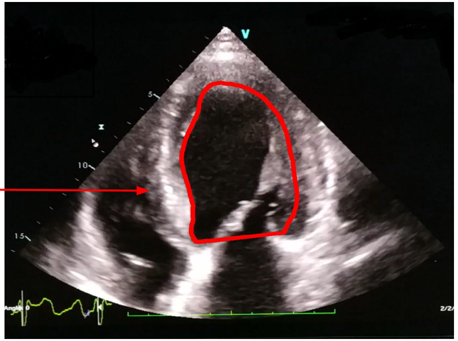
- **Problem description**: Automatic left ventricle segmentation in 2-dimensional echocardiogram images
- **Application**:
  - assessment of stroke volume, ejection fraction, wall motion and wall thickening
    → heart condition

left ventricle



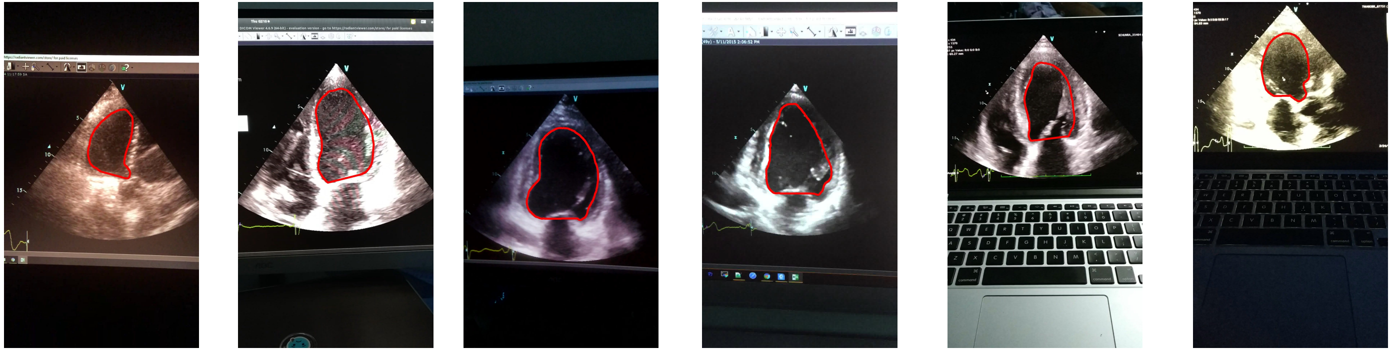
echocardiogram image

LV segmentation



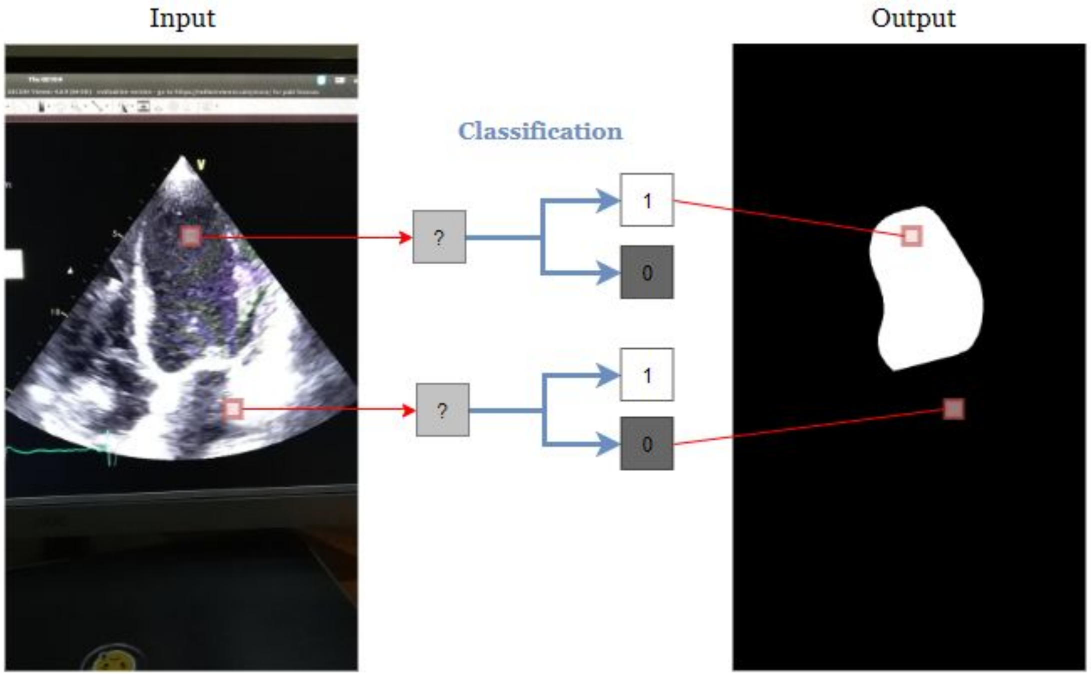
Introduction: Approach

- **Approach**: deep learning
- **Data**: echocardiogram images taken by smartphones
  $\rightarrow$ application in mobile devices
- **Difficulties**: echocardiogram noise, environmental noise, lighting variability, camera variability



Problem Formalization

- **Input:**
  3D array I = (x_{ij}) ∈ N^{HxWx3}
  (RGB image)
- **Goal:** classify x_{ij} to LV (1) or not LV (0)
- **Output:**
  binary matrix M ∈ $\{0,1\}^{HxW}$
  (mask)

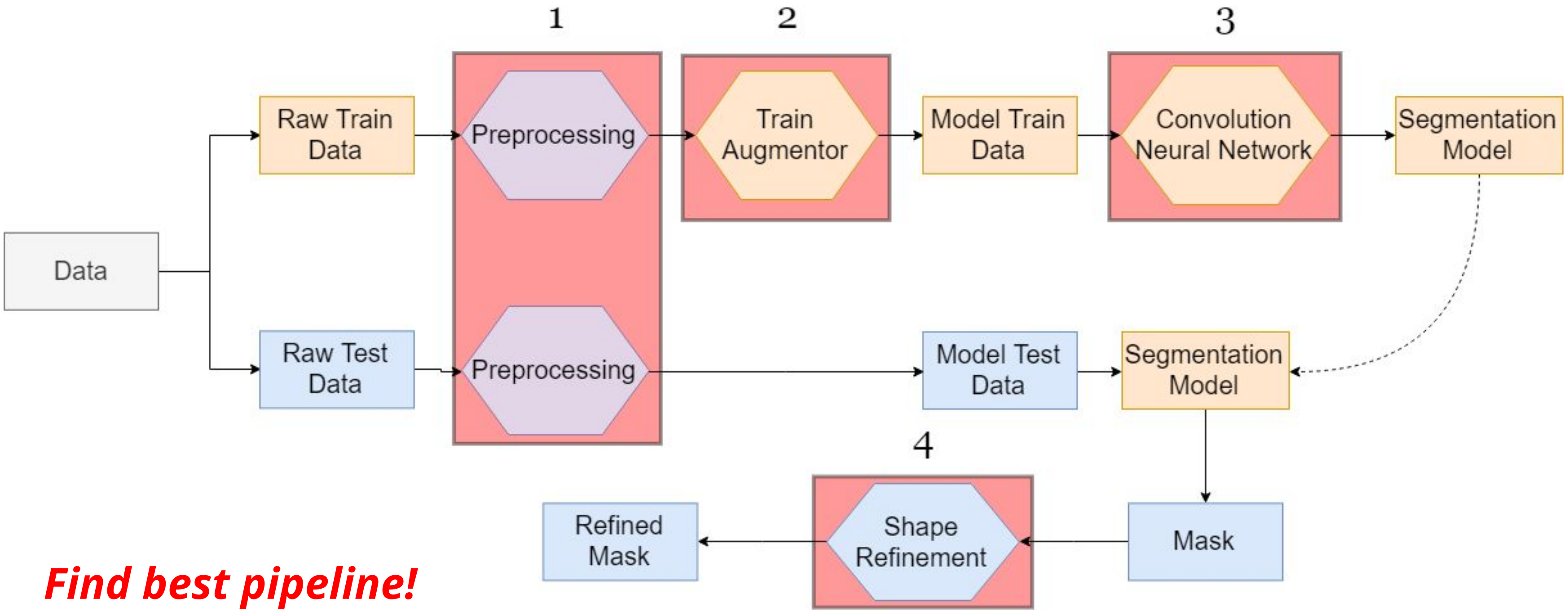


Related Work

- **Traditional approach**:
  - active contours
  - level sets
  - active shape/appearance models
  - Kalman filter
- **Deep learning approach**:
  - Smistad (2017): Unet CNN on pseudo dataset
  - Smistad (2018): Unet CNN for each heart view in echocardiogram
  - Arafati (2018): FCN-VGG architecture to segment all chambers.

⇒ promising results

Pipeline

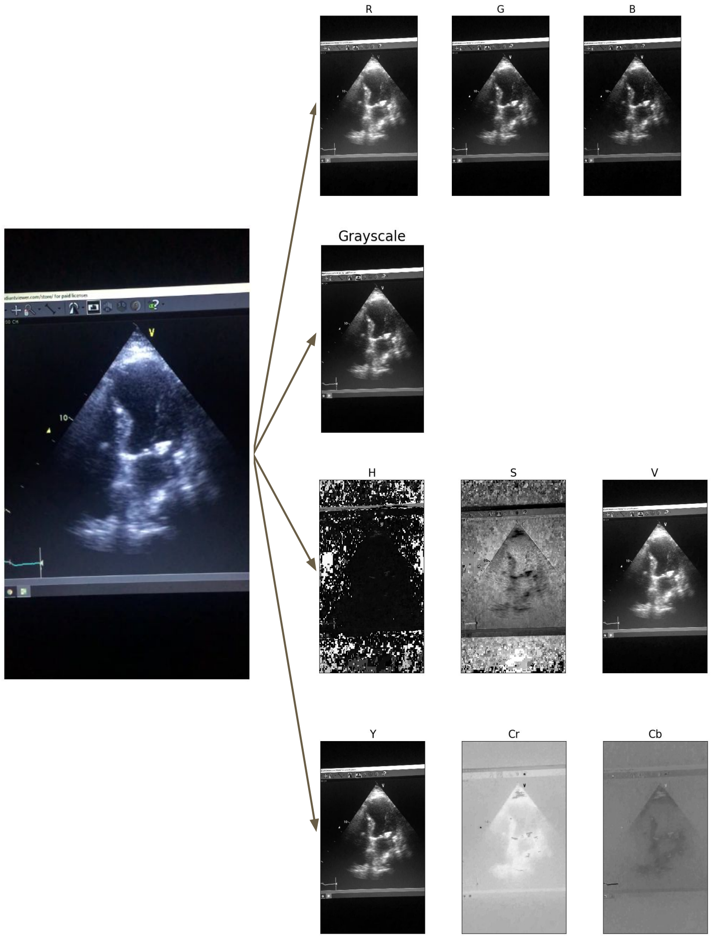


Find best pipeline!

Preprocessing: Color space

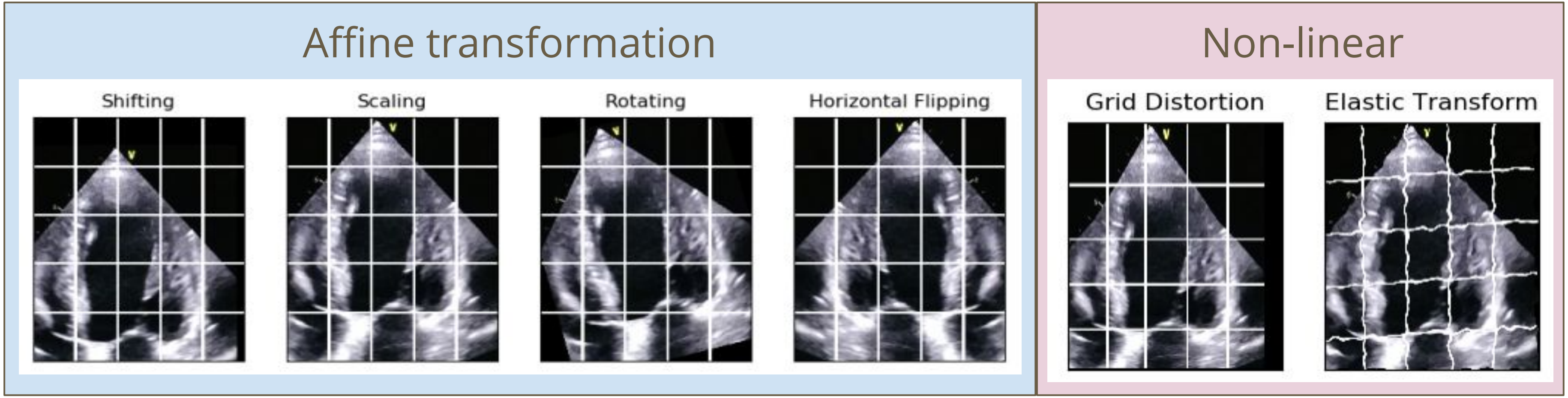
Four color spaces were experimented:

- RGB: red, green, blue
- Grayscale: only intensity
- HSV: hue, saturation, value
  - H, S: color component
  - V: color brightness
- YCbCr:
  - Y: color brightness
  - Cb, Cr: color component



Preprocessing: Data Augmentation

- Data augmentation is a technique to create new data from available data.
- Use 6 transformations: 4 *linear* (affine) and 2 *non-linear*.



Loss function

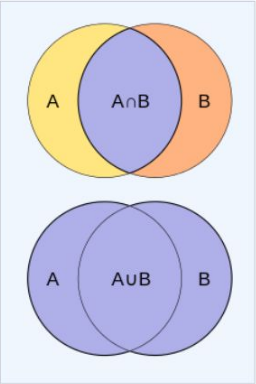
$$\textbf{Binary cross-entropy loss} = t\log p + (1-t)\log(1-p) = \begin{cases} \log p & \text{if } t = 1 \\ \log(1-p) & \text{if } t = 0 \end{cases}$$

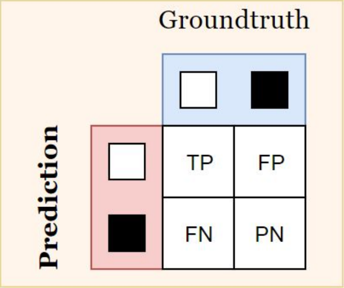
$$\textbf{Jaccard loss} = 1 - \frac{|A \cap B|}{|A \cup B|} = 1 - \frac{TP}{TP + FP + FN}$$

$$\textbf{Dice loss} = 1 - \frac{2|A \cap B|}{|A| + |B|} = 1 - \frac{2TP}{2TP + FP + FN}$$

combine losses

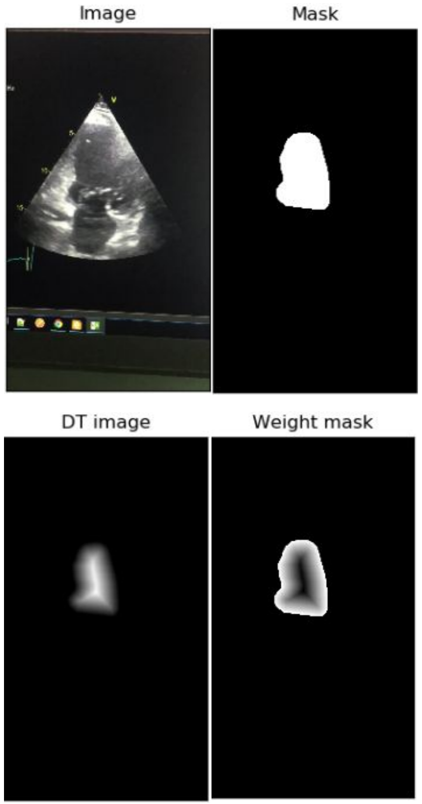
$$\textbf{BCE Jaccard} = \alpha \times \text{Binary cross-entropy loss} + \text{Jaccard loss}$$





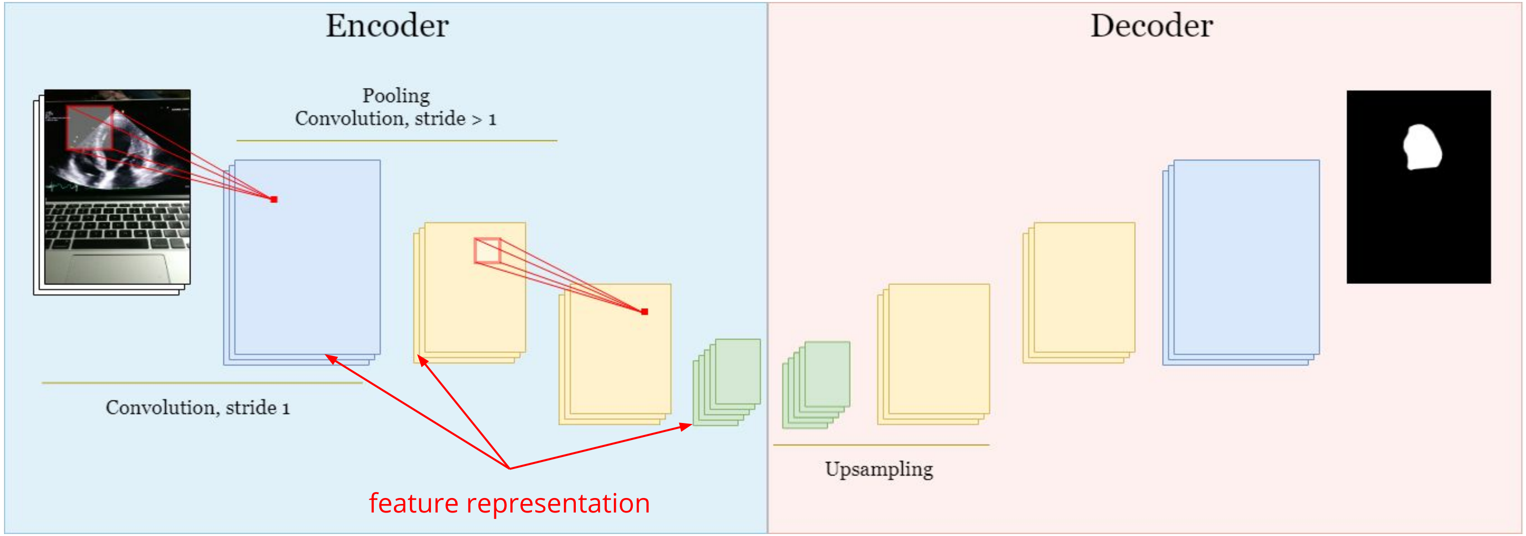
Loss function: Boundary-weighted loss

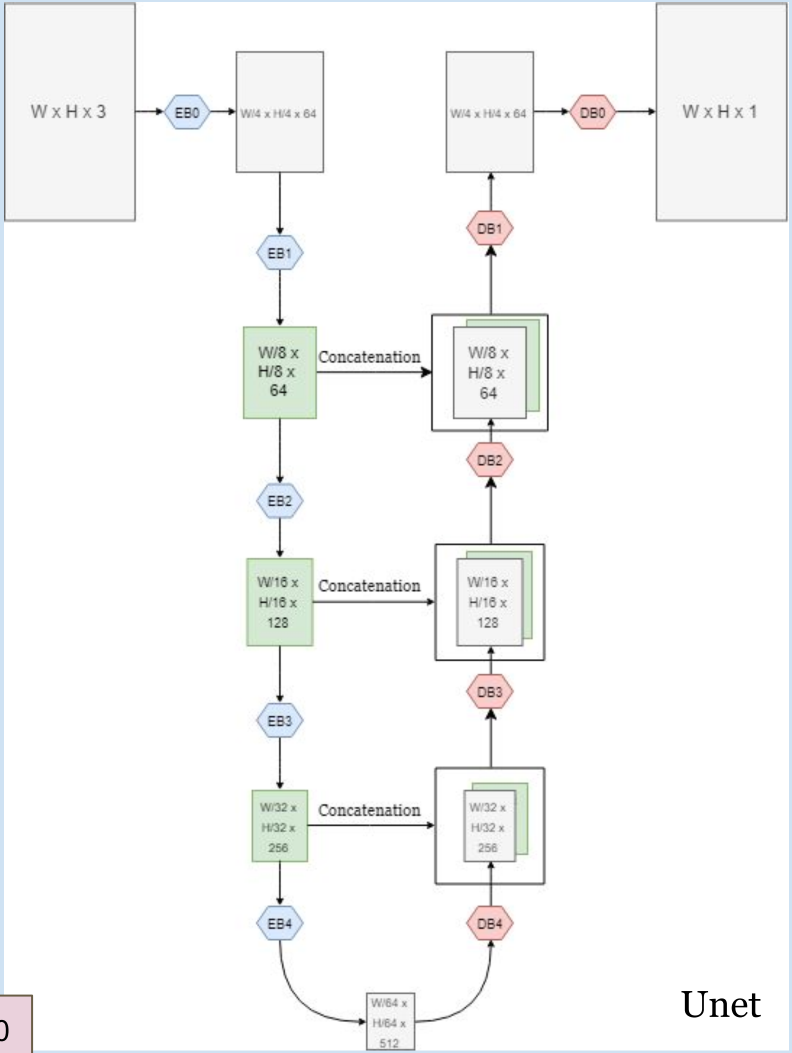
- Misclassifications usually happen in pixels near object boundary.
  → more penalties on these misclassifications
- Pixel weight assignment:
  - outside LV = 0
  - boundary (max) → center (min)
- These weights can be generated automatically using the *distance transform* (DT).



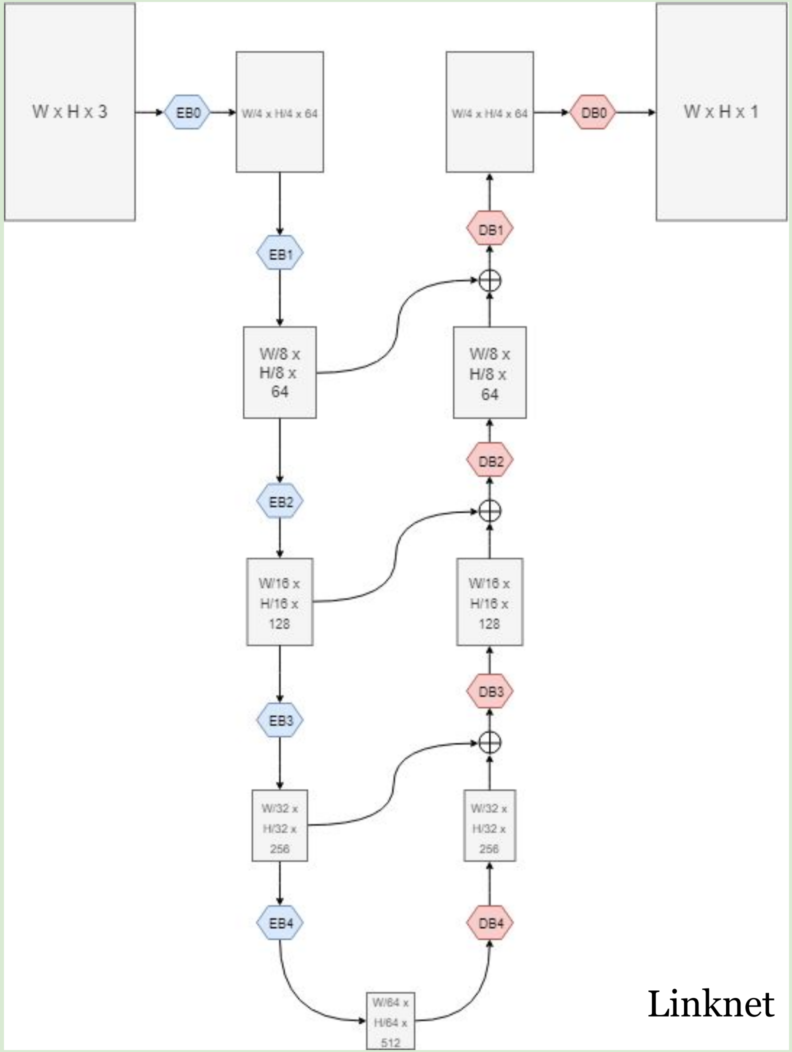
Architectures: Encoder-Decoder CNN

- **Architecture**: encoder-decoder based on convolution neural network (CNN).
- **Encoder**: input space → feature space
  Decoder: feature space → input space





Unet

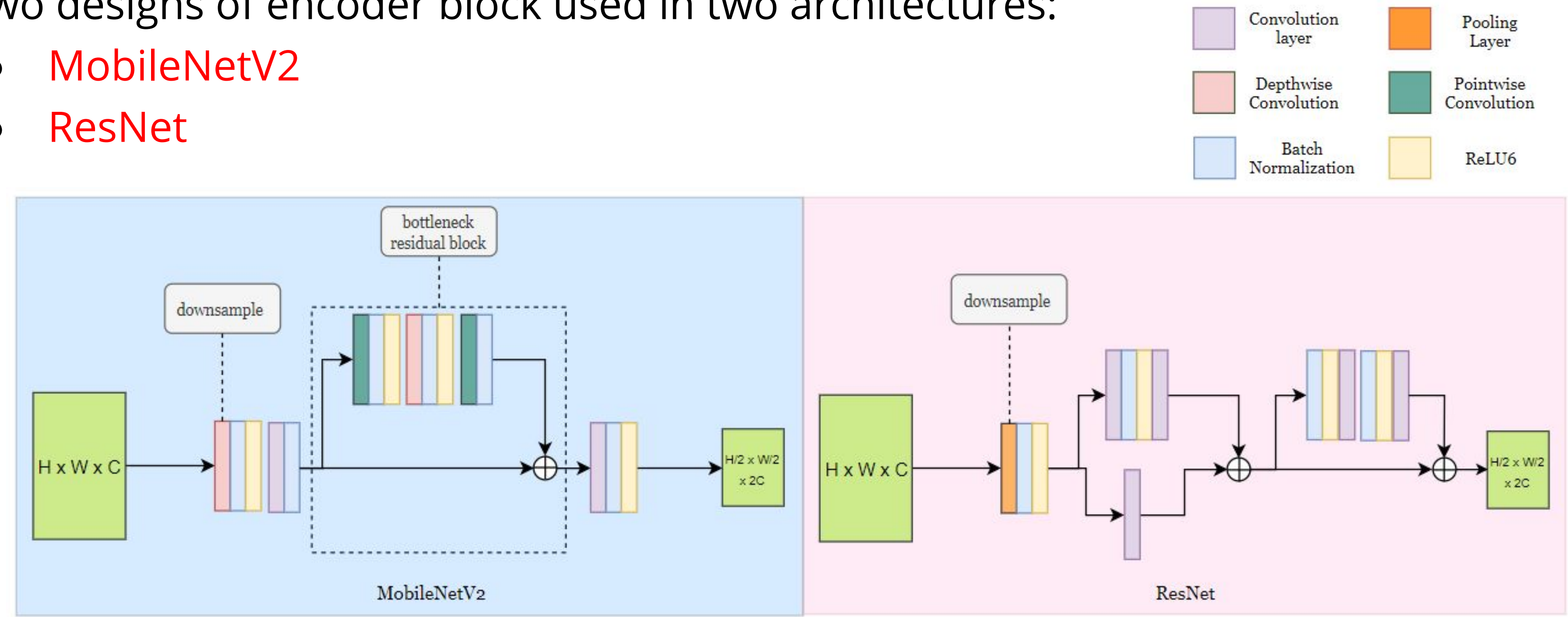


Linknet

Encoders: Blocks

Two designs of encoder block used in two architectures:

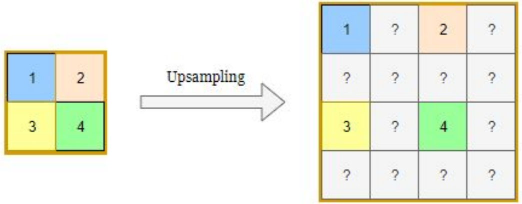
- MobileNetV2
- ResNet

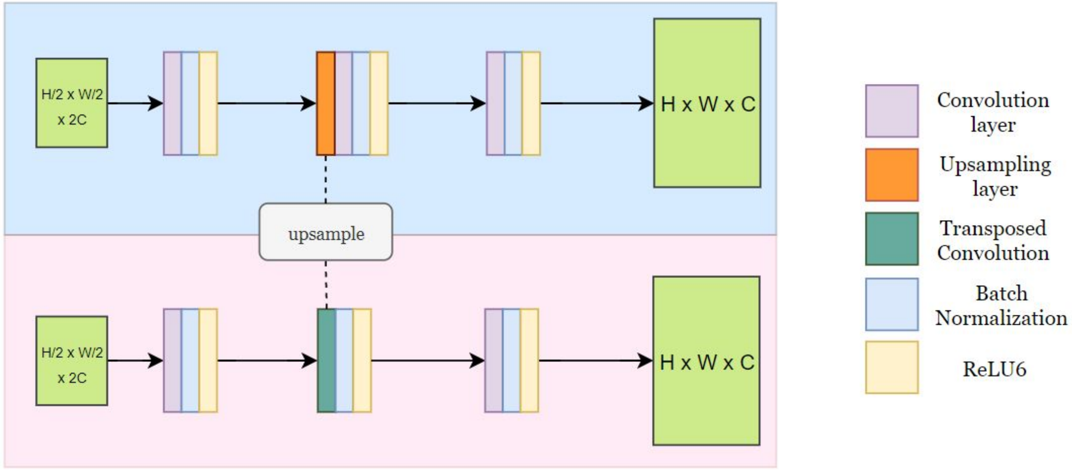


Decoder: Blocks

Two designs of decoder block based on two upsampling methods:

- Nearest neighbors
- Transposed convolution

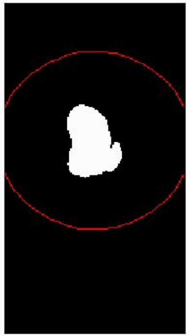




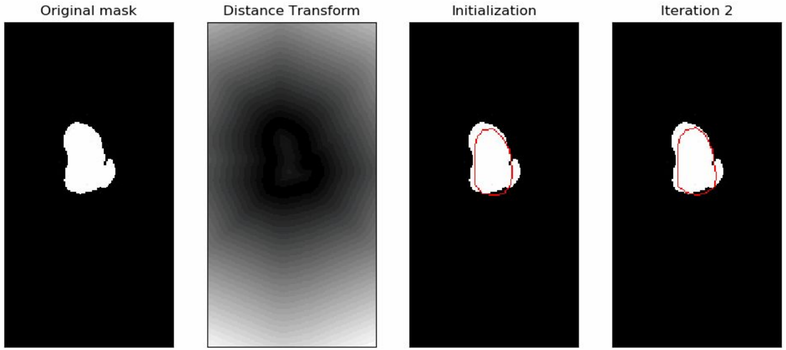
Postprocessing: Active models

- **Motivation**: *abnormal shape* of segmentation result
- **Two approaches**:
    - Active contour model: fits a contour onto the object
    - Active shape model:
        - represents shapes of an object by a statistical model
        - uses the model to fit a shape onto an object instance in a new image

Active contour model



Active shape model

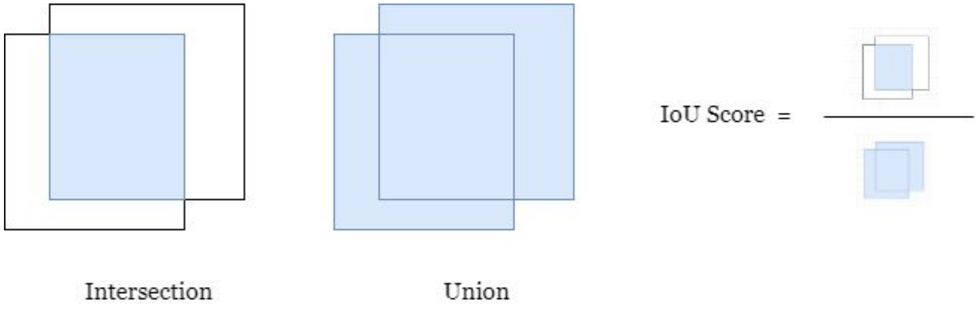


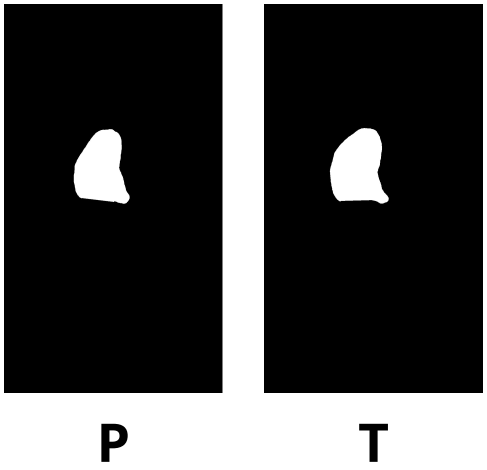
Experiment details

- *Limited computational resources*
  → not all combinations of methods could be tested.
- Experiments were done method after method:
  - Color space → Data augmentation → Loss function → Architecture → Postprocessing.
- Details:
  - **Data**: pairs of (image, mask) annotated by an expert.
    - Total: 2418 (*train*: 1909, *test*: 509, ratio: 8:2)
  - **Computational resource**: *Google Colab* (Tesla K80 GPU)
  - **Implementation**:
    - CNN architecture: *Tensorflow Keras*
    - ACM: *scikit-image*
    - ASM: *Pytorch*
  - **CNN optimization**: RMSprop, learning rate = 0.001

Metric: Intersection-over-Union (IoU) score

$$\mathrm{IoU}(P,T) = \frac{\text{Area of overlap}}{\text{Area of uinon}} = \frac{\sum_{ij}(p_{ij} \text{ and } t_{ij})}{\sum_{ij}(p_{ij} \text{ or } t_{ij})}$$





Result: Preprocessing

Mean IoU

Standard deviation IoU

Color model	mIoU	stdIoU
RGB	0.8462	0.0916
HSV	**0.8501**	**0.0797**
YCbCr	0.8498	0.0831
Grayscale	0.8466	0.0908

- Best = HSV (*YCbCr* follows closely).
- Hypothesis: they seperate brightness component
  → model adapting to changes in lighting condition

- *Horizontal flipping* improves performance.
- Best = affine (*linear*) + grid distortion (*non-linear*)

Augmentation	mIoU	stdIoU
Shift, Scale, Rotate	0.8648	0.0828
Shift, Scale, Rotate, Hflip (Affine)	0.8714	0.0711
Grid Distortion	0.8587	0.0897
Elastic Transform	0.8582	0.0956
Affine, Grid Distortion	**0.8744**	**0.0710**

Result: Loss function and Architecture

Loss function	mIoU	stdIoU
BCE	0.8651	0.0827
Dice	0.8737	0.0676
BCE Dice	0.8726	0.0736
Jaccard	0.8726	0.0637
BCE Jaccard	0.8744	0.0710
Weighted BCE Jaccard	**0.8805**	**0.0633**

- Test *non-weighted* losses
  → best = *BCE Jaccard*
- Test weighted BCE Jaccard
  → improvement

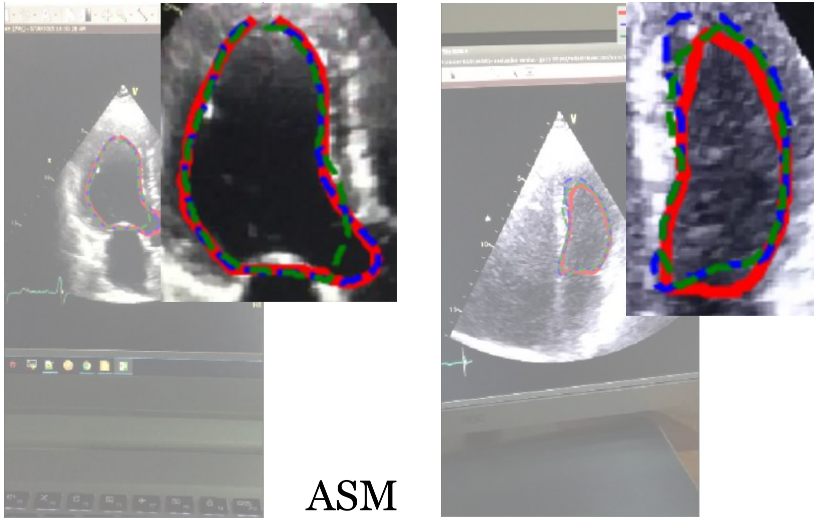
- Fix **architecture** = *Linknet*, **encoder** = *MobileNetV2*, test **decoder** ∈ {Upsampling, Transpose}
- Fix **decoder** = *Upsampling*, test **architecture** ∈ {Linknet, Unet}, **encoder** ∈ {MobileNetV2, ResNet}

Architecture	Encoder	Decoder	mIoU	stdIoU
Linknet	MobileNetV2	Upsampling	**0.8805**	**0.0633**
Linknet	MobileNetV2	Transpose	0.8753	0.0846
Linknet	ResNet	Upsampling	0.8720	0.0685
Unet	MobileNetV2	Upsampling	0.8737	0.0673
Unet	ResNet	Upsampling	0.8712	0.0706

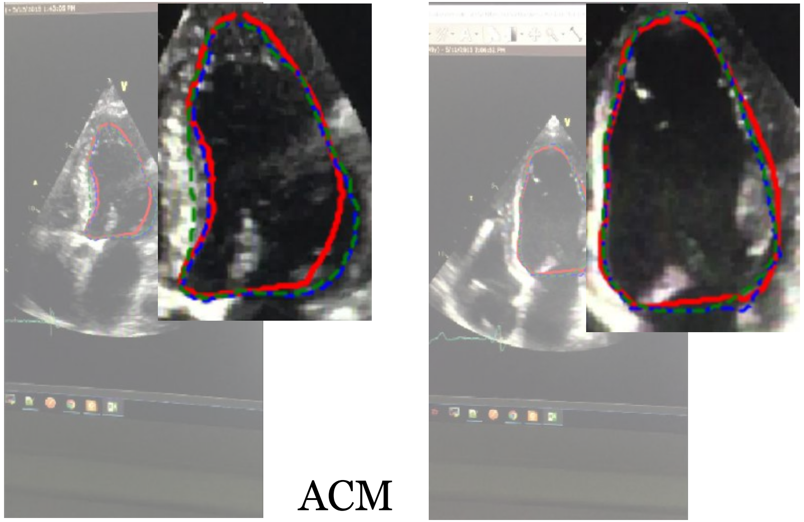
Result: Postprocessing

Postprocessing	mIoU	stdIoU
Active contour model	**0.8942**	**0.0589**
Active shape model	0.8517	0.0694

- **ACM**: slight enhancement
- **ASM**: degradation due to high variance in shapes



ASM



ACM

Conclusion

- **Best pipeline**:
  - *Color space*: HSV
  - *Data augmentation*: affine transformation and grid distortion
  - *Loss function*: weighted BCE Jaccard
  - *Architecture*: Linknet with MobileNetV2 encoder and Nearest Neighbors Upsampling decoder
  - *Postprocessing*: active contour model
- **Future work**:
  - More data
  - Echocardiogram noise simulation
  - Advanced active shape model

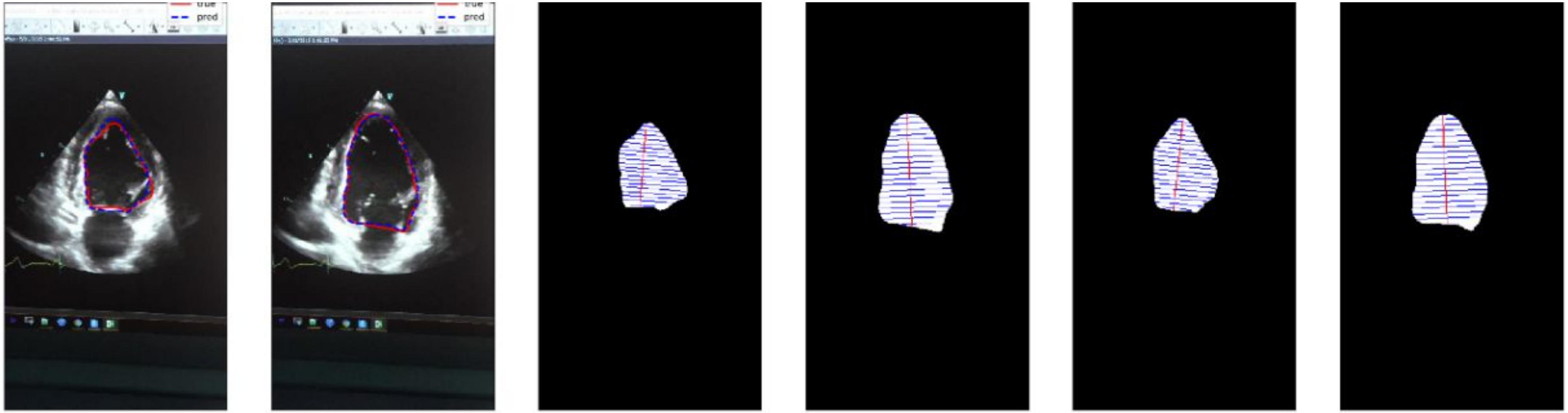
Thanks for listening!

Ejection Fraction Calculation

The volume of left ventricle can be approximated using Simpson's rule:

$$V = 0.85\,\frac{A^2}{L}, \text{ where } A = \frac{L}{3 \times 20}(a_1 + a_{20} + 4a_2 + 2a_3 + 4a_4 + 2a_5 + \cdots + 2a_{19}).$$

Then EF can be computed as: $EF = 1 - \frac{V_{min}}{V_{max}}$.

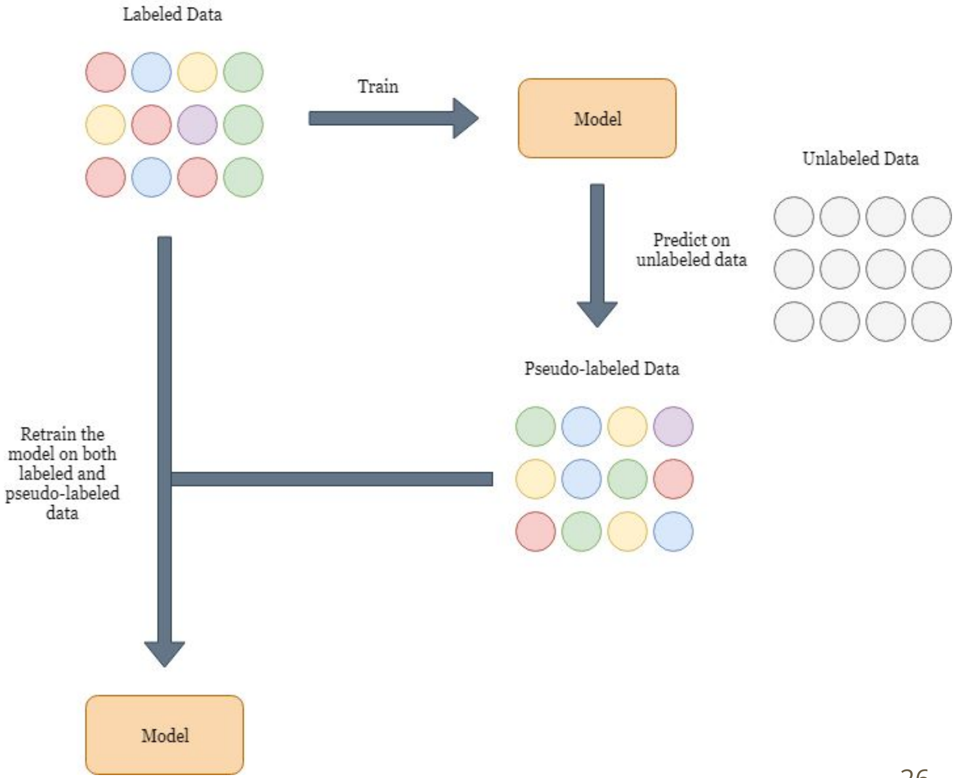


Technique: Pseudo labeling

- Pseudo labeling is a ***semi-supervised learning*** technique, i.e. a technique that also makes use of ***unlabeled data*** for training along with labeled data.

- It uses a ***trained model*** to get predictions on unlabeled data instances, then add instances with pseudo-generated labels of high confidence to the current training set, and retrain a model on the new dataset.

- Intuitively, this technique encourages model to make confident predictions on unlabeled data.

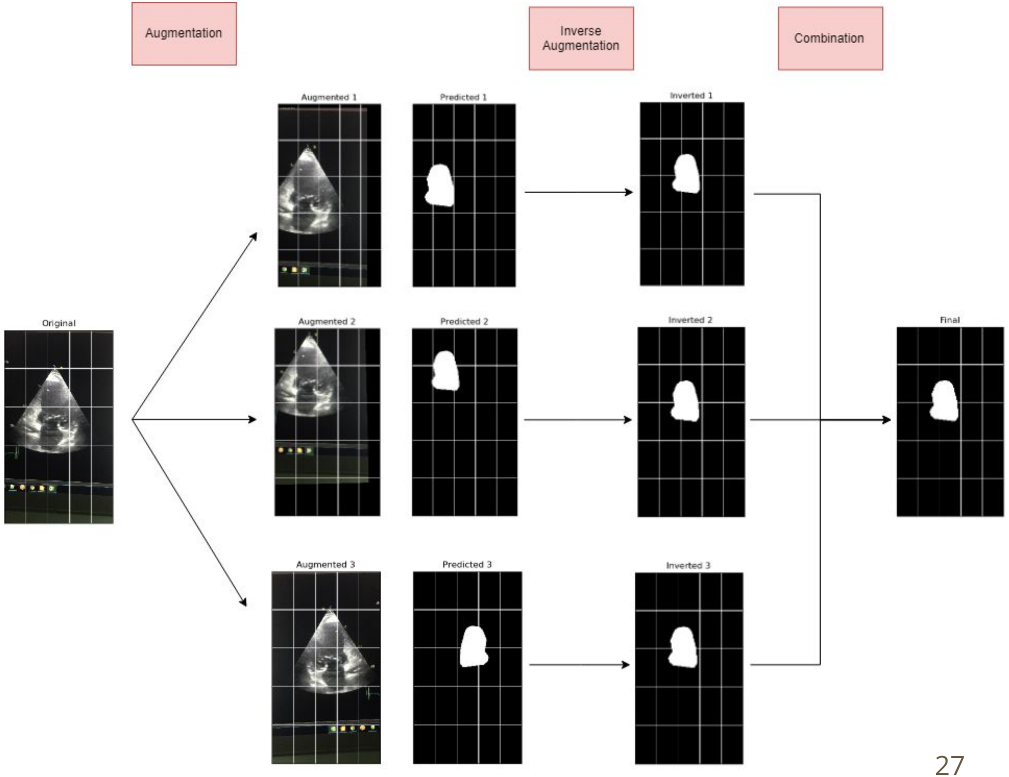


Postprocessing: Test-time augmentation

- Get predictions on differently transformed versions of an image, then combine the results.
- **Hypothesis**: This gives more opportunities to make inference based on various views of a same image.
- Cons: computation costs.

Combination methods

Method	Pixels			Result
averaging	0.9	0.1		0.5
logical and	■	□		■
	□	□		□
	■	■		■
majority voting	■	□	■	■
	□	□	■	□

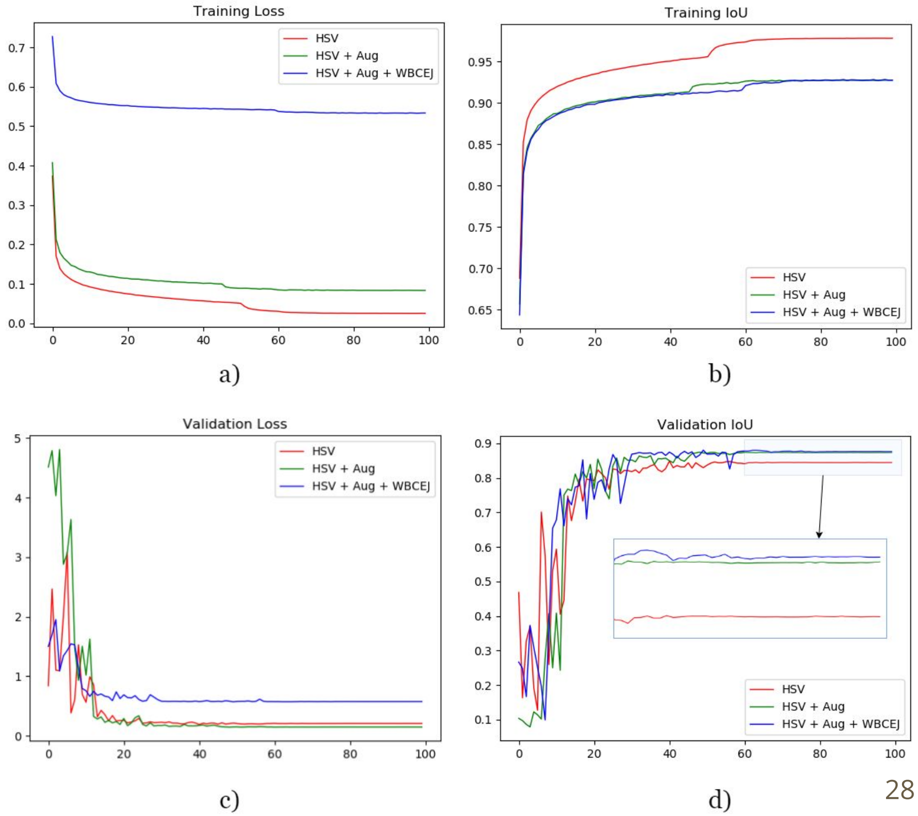


Training history

- ***HSV*** model suffers a heavy overfitting, as data which it was trained on has low variability.

- ***HSV + Aug*** model shows a considerable improvement from ***HSV*** model, indicating that data augmentation is crucial for deep learning.

- ***HSV + Aug + WBCEJ*** gives a slightly better IoU score on the validation set than ***HSV + Aug***, suggesting that weighted loss has positive effect on the training.



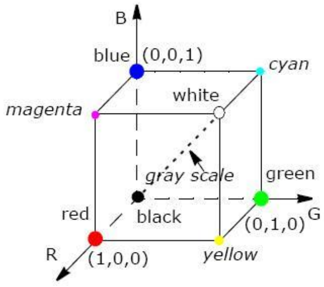
Preprocessing: Color space

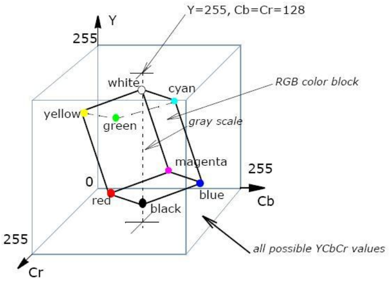
Four colors spaces were experimented:

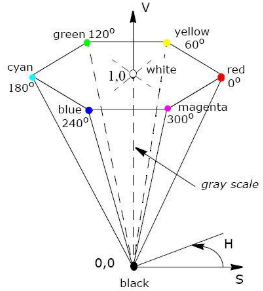
- **RGB** (red, green, blue): is similar to how human visual system works.
- **Grayscale**: carry only the intensity information.
- **HSV** (hue, saturation, value): is designed to approximate the way humans perceive and interpret color.

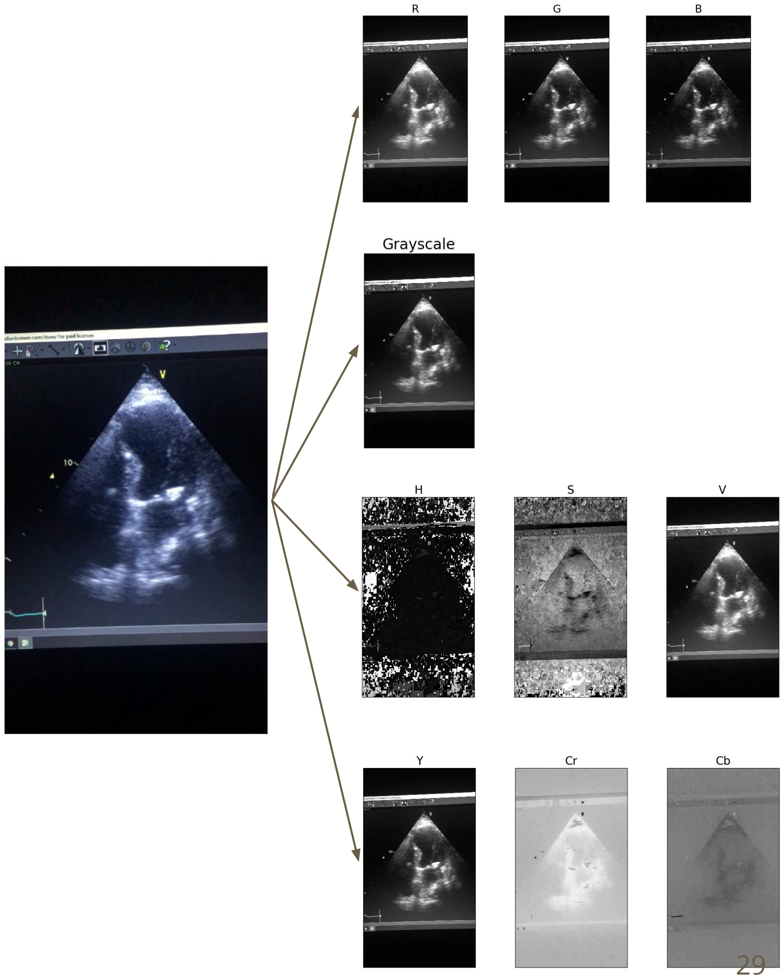
  Hue ~ color, saturation ~ $(\text{hue} \neq \text{neural})$, value ~ lightness.
- **YCbCr** (Y - luminance; Cb, Cr - chroma components):

  used in video compression, as *human eyes are more sensitive to luminance* (i.e. the brightness of the color) than chroma components (can be reduced).



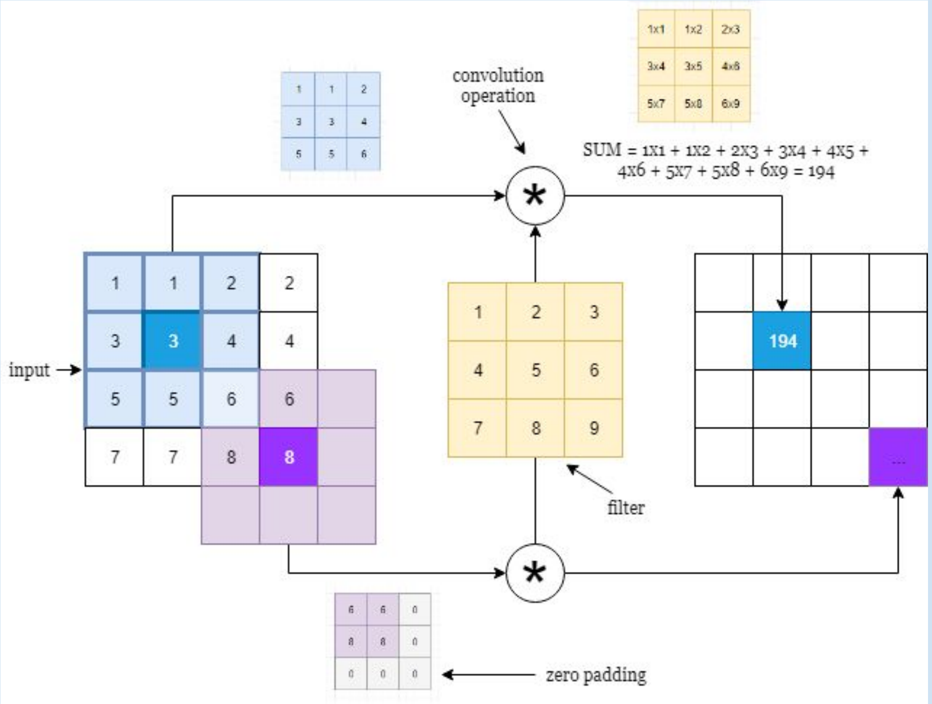




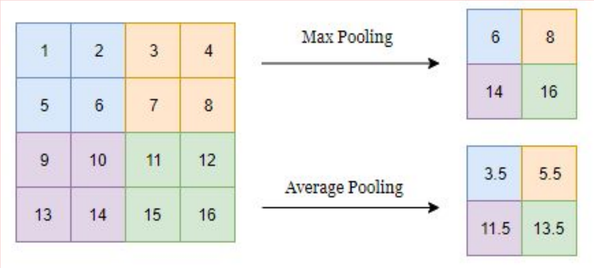


CNN Layers: Details

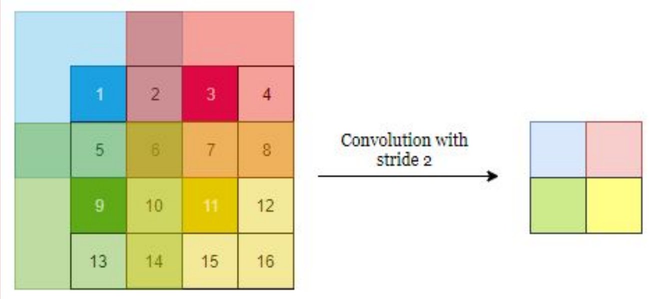
Convolution, stride 1



Pooling

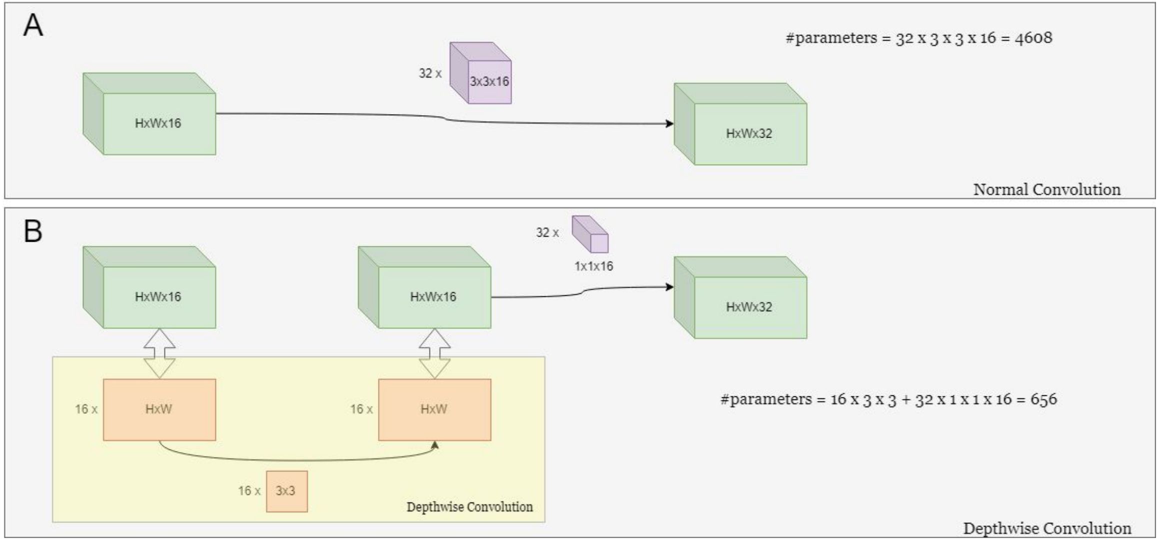


Convolution, stride 2

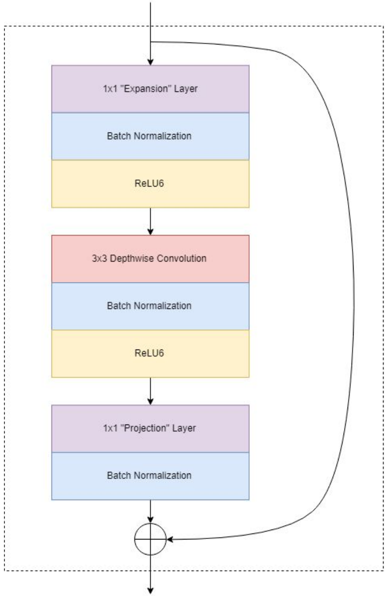


Encoders: Bottleneck Residual Block

- *Depthwise convolution block* is used in MobileNetV1 to *reduce the number of model paramterers*.
- *Bottleneck residual block* in ***MobileNetV2*** has two new components: the expansion-projection mechanism and residual connection (which was introduced in ***ResNet*** architecture).

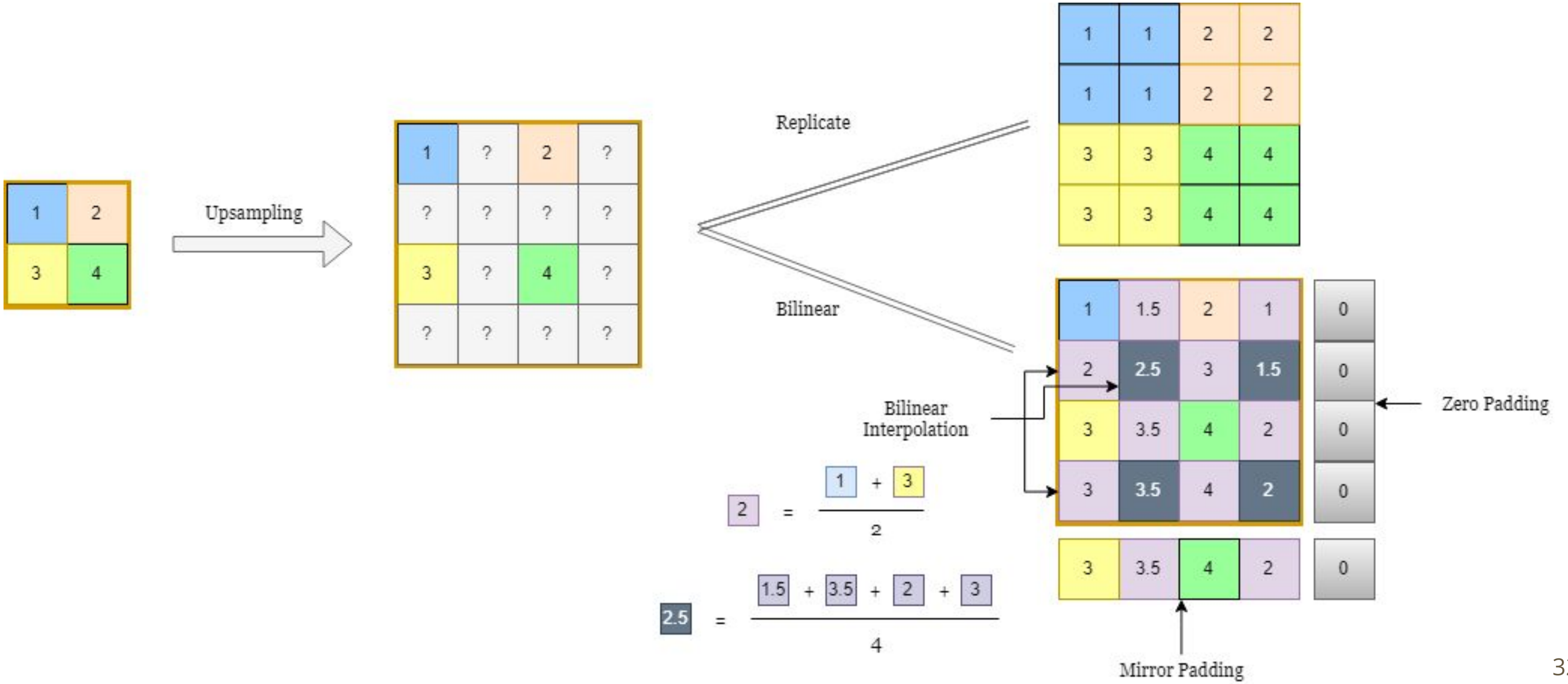


Depthwise convolution block



Bottleneck residual block

Decoder: Nearest Neighbors Upsampling



Decoder: Transposed Convolution