Mid-term Review and Final Presentation

11210IPT 553000

Deep Learning in Biomedical Optical Imaging

2023/11/20

Mid-term Review

Grading

- **Homework (35 %):** Assignments to get familiar with basic deep learning programming.
- **Midterm (30 %):** Basic concepts of deep learning and related mathematical derivation.
- **Final Project Presentation (20 %):** A project presentation of a research paper related to the deep learning application in medical imaging and the code implementation of the
- **Creativity Report (15 %):** A Provide a detailed analysis of your model implementation on a given image dataset.

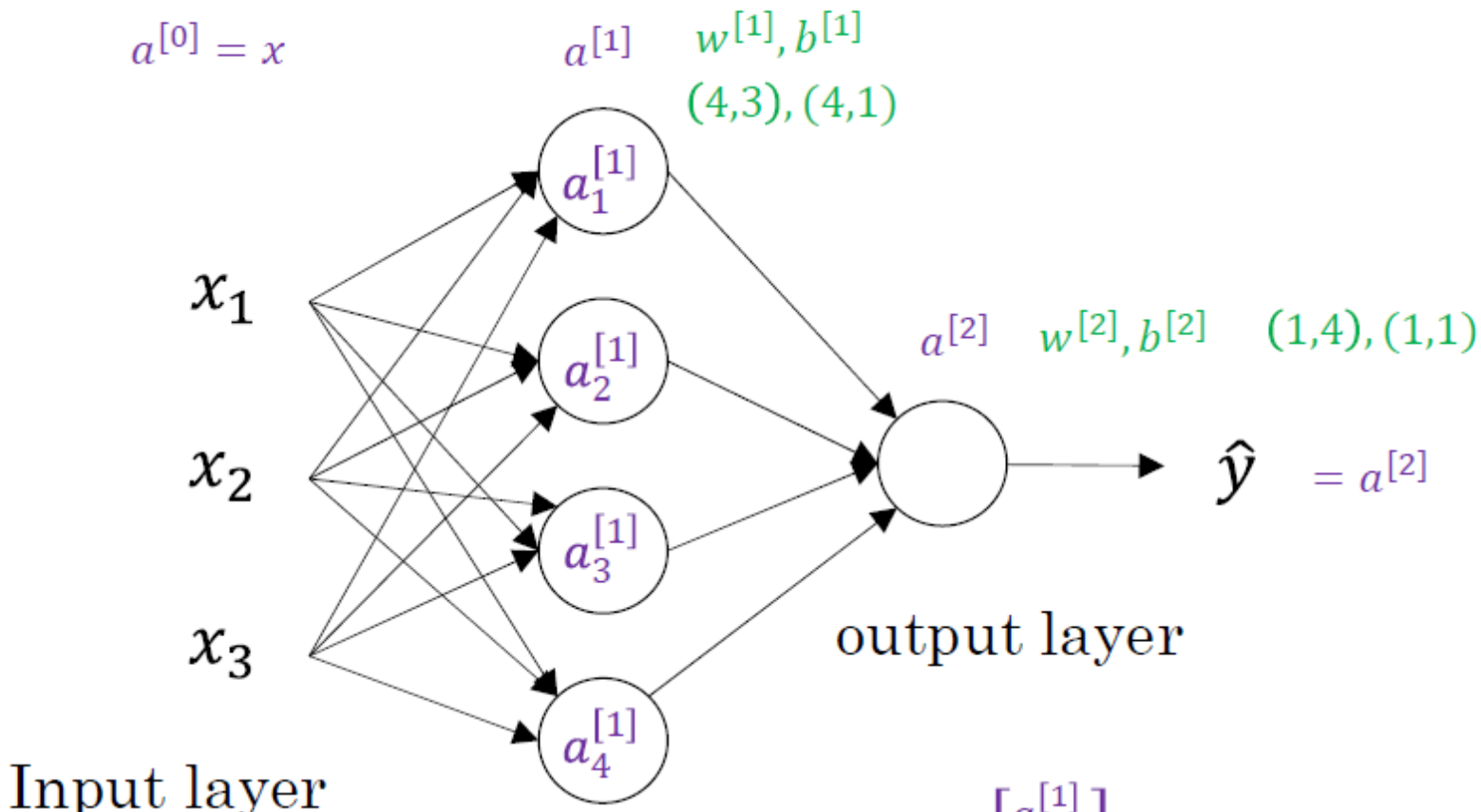
Important Concepts

- Loss Function / Cost Function
- Activation functions
- Weight Initialization
- Forward propagation
- Backward propagation
- Hyperparameters
- Parameters
- Training, validation, and test datasets
- Bias and Variance
- Overfitting
- Regularization
- Dropout
- Data augmentation
- Early stopping
- Input normalization
- Vanishing/exploding gradients
- Epoch / Batch size/ Iteration
- Learning rate settings
- Saddle point
- Optimization algorithm: Momentum / RMSprop/ Adam
- Batch Normalization
- Multiclass classification
- Cross-entropy
- Transfer Learning
- Image Kernels (Filters)
- Convolution
- Pooling
- Padding
- Stride
- Features
- CNN networks
- Unsupervised Learning
- Recurrent-based model
- Attention-based model

- NumPy API
  - How to use some functions?
  - np.zeros, xxx.shape, np.add…

One hidden layer Neural Network

Neural Network Representation

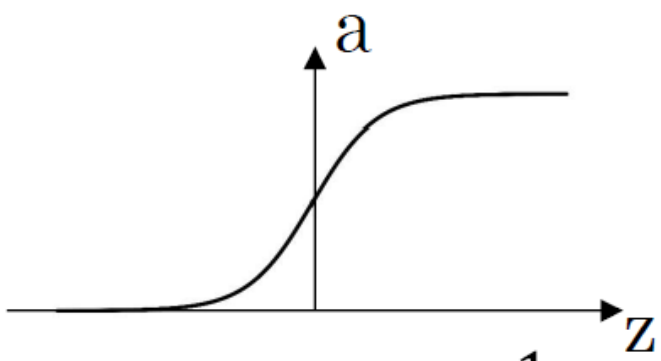


$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix}$$

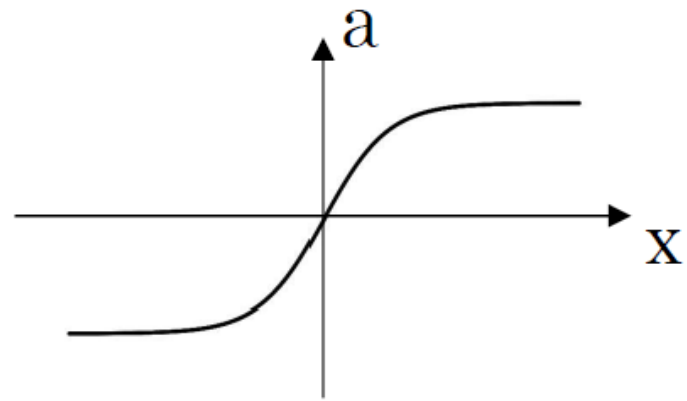
2-layer NN

Activation functions

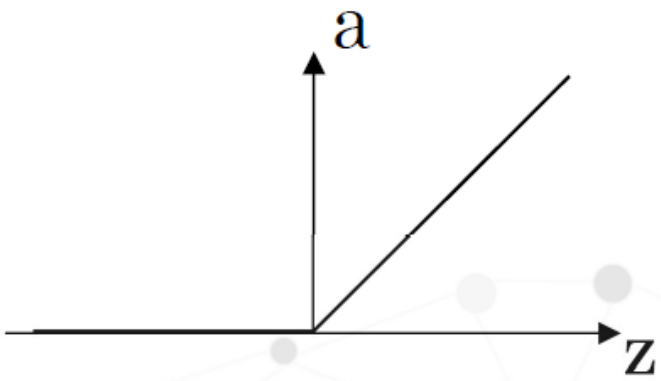
Pros and cons of activation functions



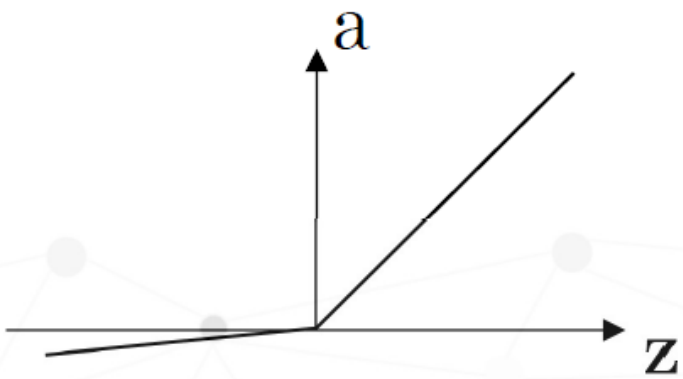
$$\text{sigmoid: } a = \frac{1}{1 + e^{-z}}$$



$$tanh: a = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

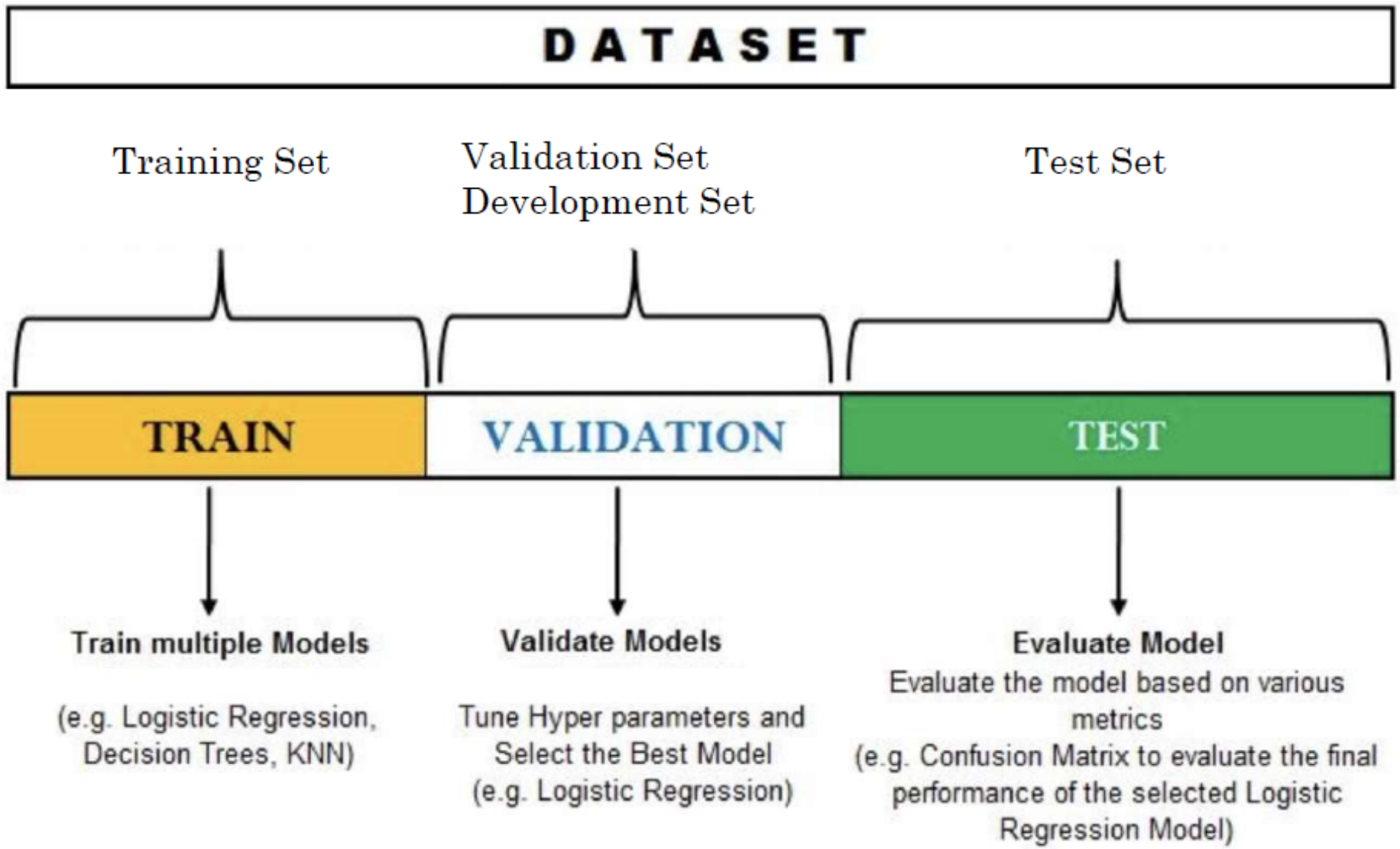


$$Relu: a = \max(0, z)$$



$$Leaky\ Relu: a = \max(0.01z, z)$$

Train/dev/test sets



Bias and Variance

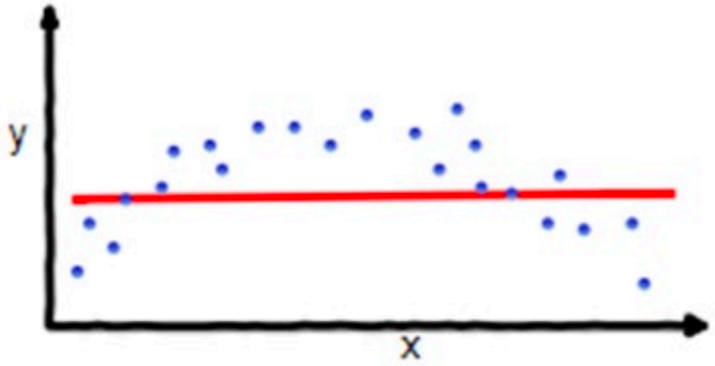
Cat classification

Train set error:	1%	15%	15%	0.5%
Dev set error:	11%	16%	30%	1%
	low bias high variance	high bias low variance	high bias high variance	low bias low variance
	overfitting	underfitting		

Human error (Bayes error): ~0%

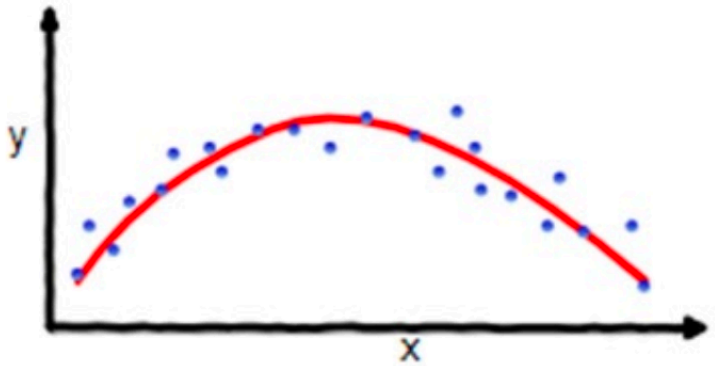
Bias and variance

Model ——

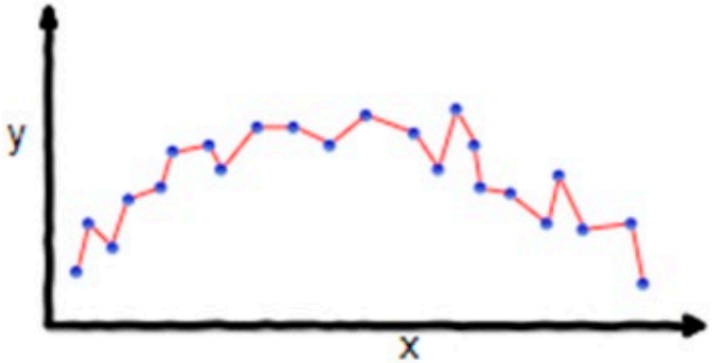


high bias

underfitting



"Just right"

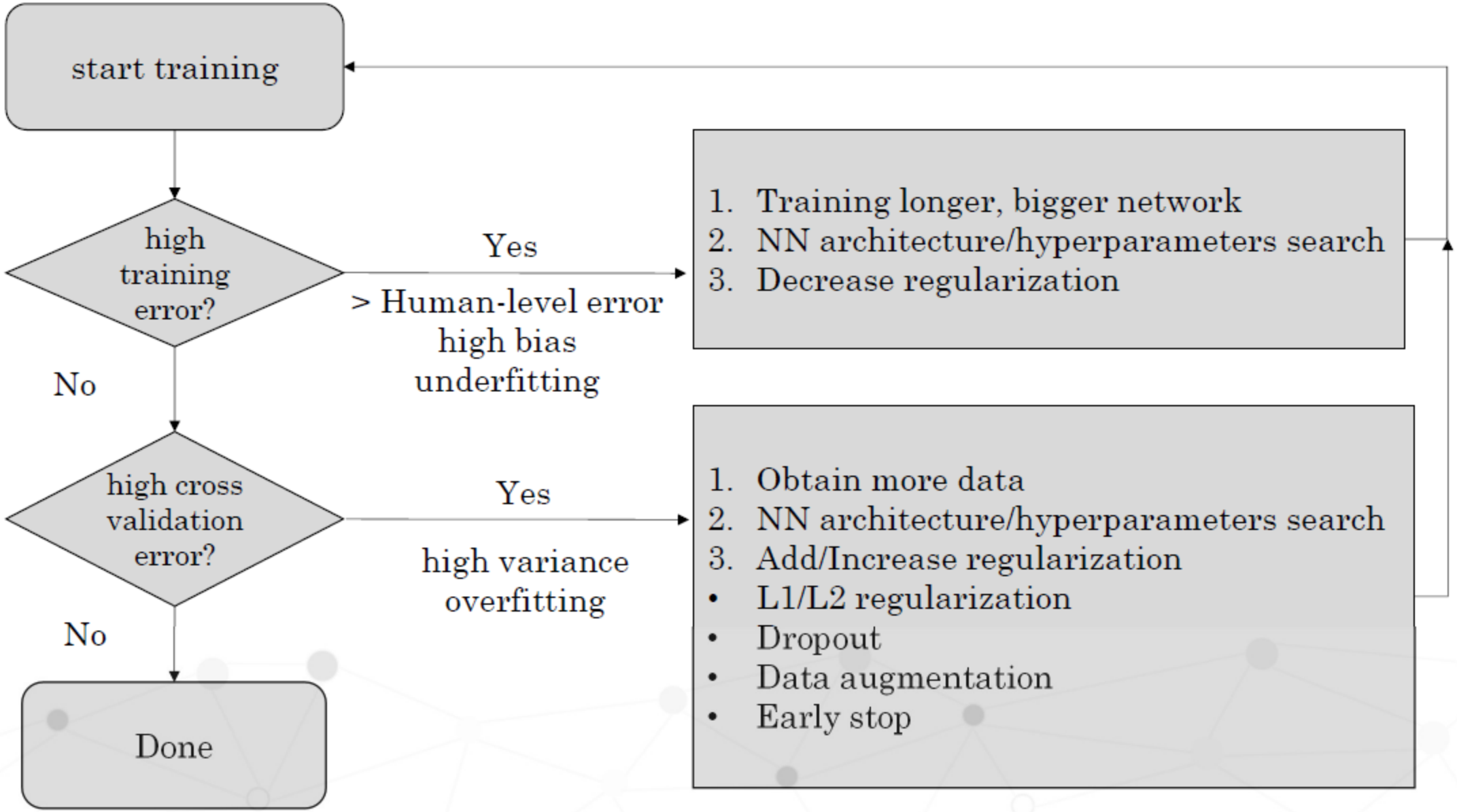


high variance

overfitting

Setting up your ML application

Basic “recipe” for machine learning



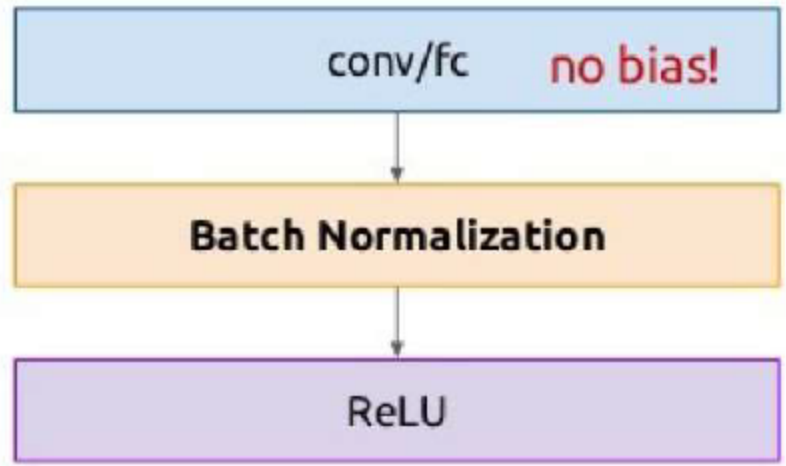
Normalizing activations in a network

Batch Normalization

Works by **re-normalizing layer inputs to have zero mean and unit standard deviation** with respect to running batch estimates.

Also adds a **learnable scale and bias** term to allow the network to still use the nonlinearity.

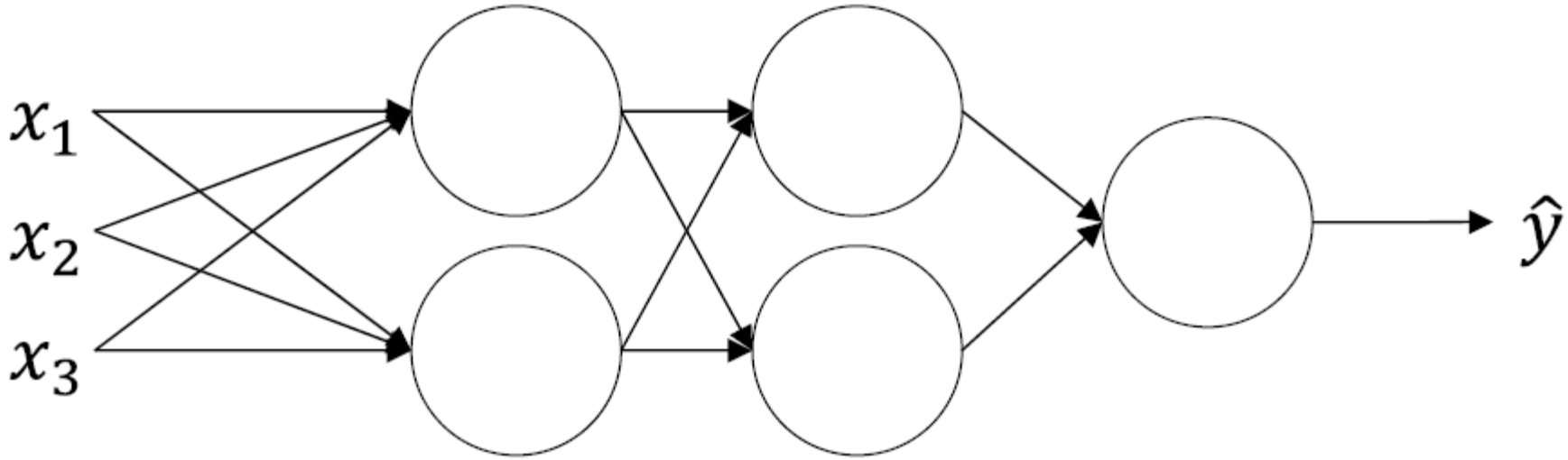
Usually **allows much higher learning rates!**



Ioffe and Szegedy. **Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift**, JMRL 2015

Fitting Batch Norm into a neural network

Adding Batch Norm to a network



$x^{[0]} \xrightarrow{W^{[1]},\, b^{[1]}} z^{[1]} \rightarrow a^{[1]} = g^{[1]}(\tilde{z}^{[1]}) \xrightarrow{W^{[2]},\, b^{[2]}} z^{[2]} \rightarrow a^{[2]} \ldots\ldots\ldots$

$x^{[0]} \xrightarrow{W^{[1]},\, b^{[1]}} z^{[1]} \xrightarrow[\text{Batch norm}]{\gamma^{[1]},\, \beta^{[1]}} \tilde{z}^{[1]} \rightarrow a^{[1]} = g^{[1]}(\tilde{z}^{[1]}) \xrightarrow{W^{[2]},\, b^{[2]}} z^{[2]} \xrightarrow[\text{Batch norm}]{\gamma^{[2]},\, \beta^{[2]}} \tilde{z}^{[2]} \rightarrow a^{[2]} \ldots\ldots\ldots$

$Parameters:\ W^{[1]},\, b^{[1]},\, W^{[2]},\, b^{[2]} \ldots\ldots,\, W^{[L]},\, b^{[L]}$

$\gamma^{[1]},\, \beta^{[1]},\, \gamma^{[2]},\, \beta^{[2]} \ldots\ldots, \gamma^{[L]},\, \beta^{[L]}$

Computer Vision

Vertical edge detection

3X1+1X1+1X2+0X0+0X5+0X7+1X-1+8X-1+2X-1

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6 X 6

*

Convolution

Filter or Kernel

1	0	-1
1	0	-1
1	0	-1

3 X 3

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4 X 4

Strided Convolutions

Output dimension

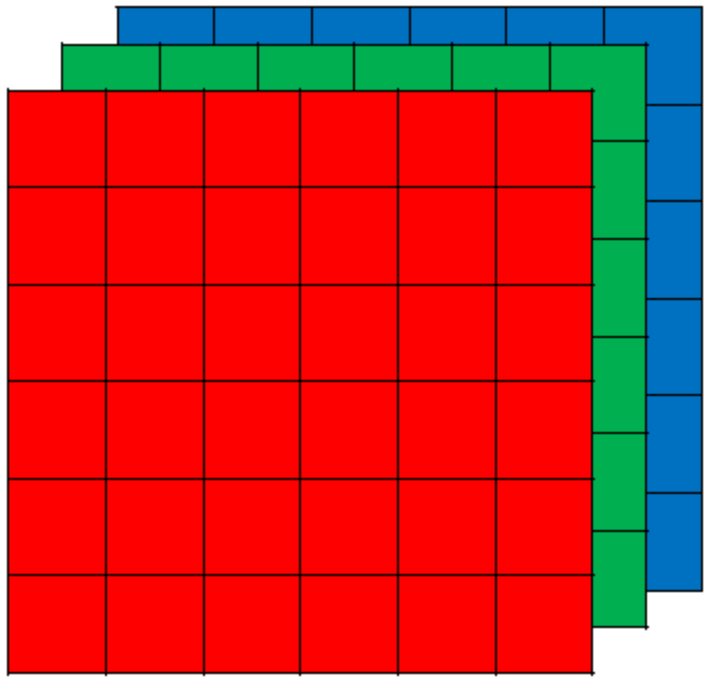
$n \times n$ image $\quad f \times f$ filter $\quad$ padding p $\quad$ stride s

Output size

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

$\lfloor x \rfloor = floor\ (x)$

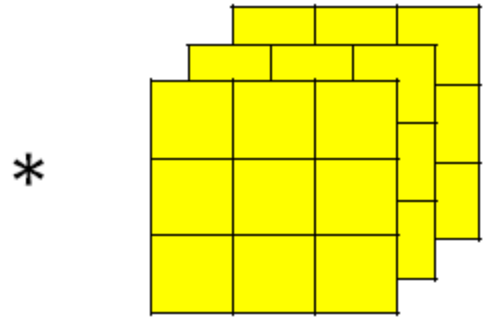
Convolution Over Volumes



6 x 6 x 3

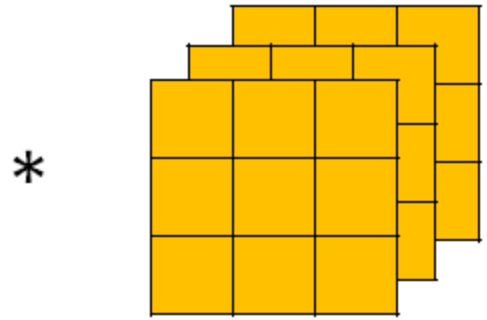
Multiple filters

Vertical edge

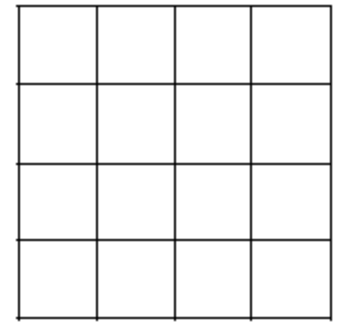


3 x 3 x 3

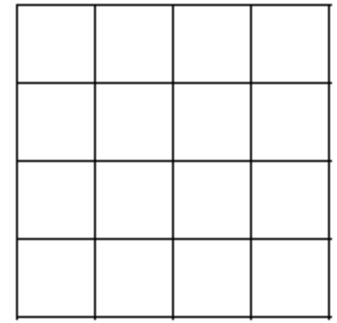
Horizontal edge



3 x 3 x 3

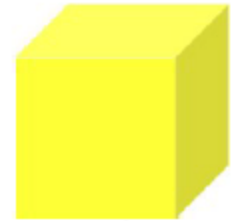


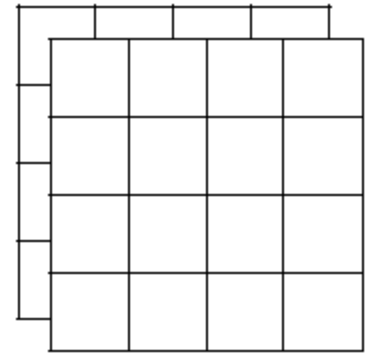
4 x 4



4 x 4

4 x 4 x 2



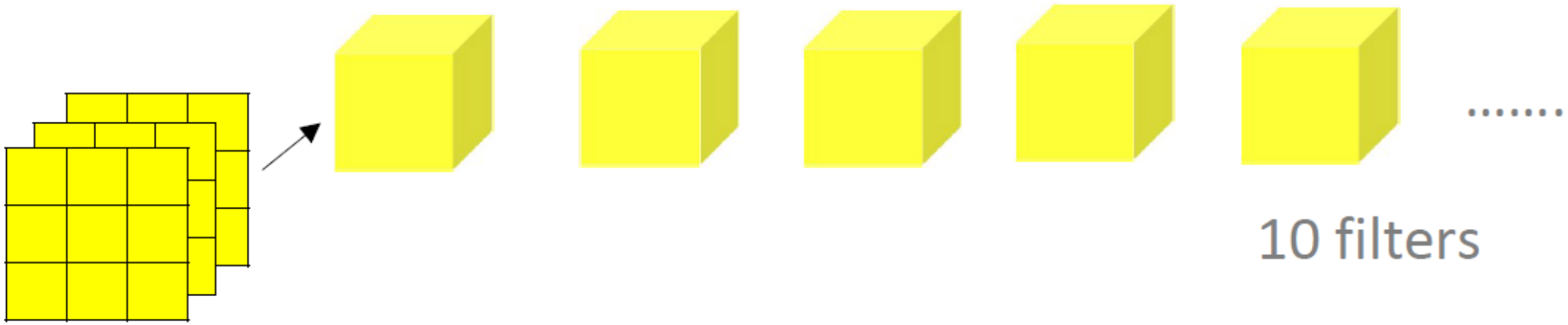


# of filters

Summary: $n \times n \times n_c * f \times f \times n_c \rightarrow (n-f+1) \times (n-f+1) \times n'_c$

Number of parameters in one layer

If you have 10 filters that 3 x 3 x 3 in one layer of a neural network, how many parameters does this layer have?



27 parameters + 1 bias
= 28 parameters

28 X 10 = 280 parameters

CNN Model

CONV2D

Conv2d
Conv2d

CLASS torch.nn.Conv2d(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *padding_mode='zeros'*, *device=None*, *dtype=None*) [SOURCE] https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html

Applies a 2D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

where $\star$ is the valid 2D cross-correlation operator, N is a batch size, C denotes a number of channels, H is a height of input planes in pixels, and W is width in pixels.

This module supports TensorFloat32.

On certain ROCm devices, when using float16 inputs this module will use different precision for backward.

- `stride` controls the stride for the cross-correlation, a single number or a tuple.
- `padding` controls the amount of padding applied to the input. It can be either a string {'valid', 'same'} or an int / a tuple of ints giving the amount of implicit padding applied on both sides.

CNN Model

MAXPOOL2D

MaxPool2d
MaxPool2d

CLASS torch.nn.MaxPool2d(*kernel_size*, *stride=None*, *padding=0*, *dilation=1*, *return_indices=False*, *ceil_mode=False*) [SOURCE]

Applies a 2D max pooling over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C, H, W), output (N, C, H_{out}, W_{out}) and `kernel_size` (kH, kW) can be precisely described as:

$$out(N_i, C_j, h, w) = \max_{m=0,\ldots,kH-1} \max_{n=0,\ldots,kW-1} \text{input}(N_i, C_j, \text{stride}[0] \times h + m, \text{stride}[1] \times w + n)$$

If `padding` is non-zero, then the input is implicitly padded with negative infinity on both sides for `padding` number of points. `dilation` controls the spacing between the kernel points. It is harder to describe, but this link has a nice visualization of what `dilation` does.

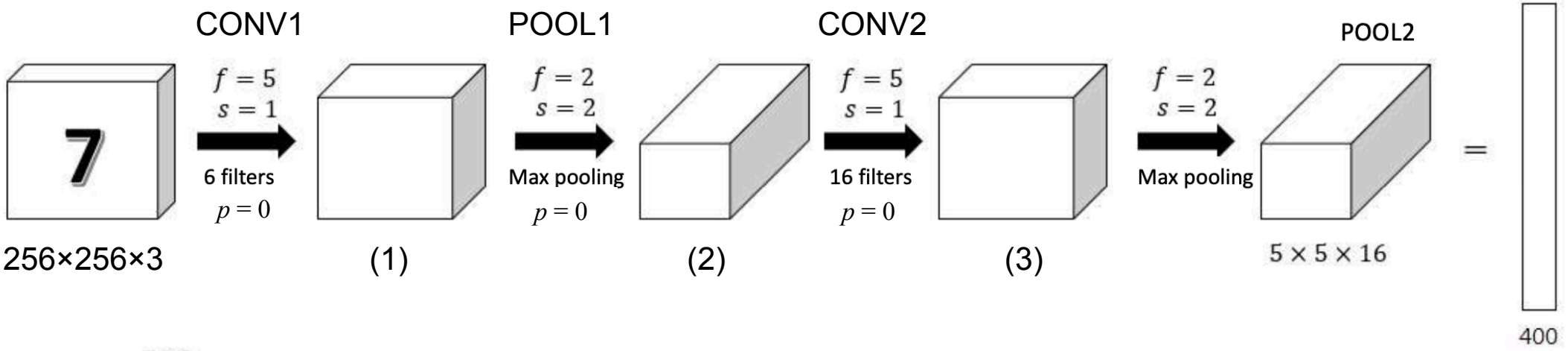
https://pytorch.org/docs/stable/generated/torch.nn.MaxPool2d.html

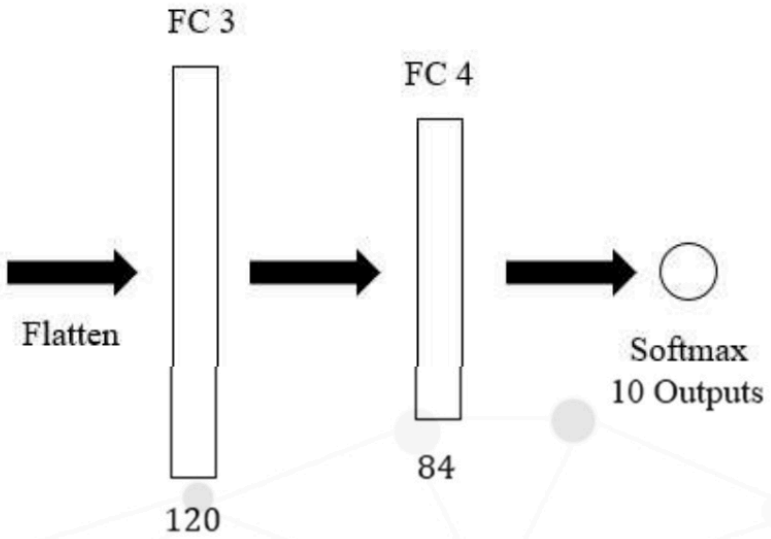
NOTE

When ceil_mode=True, sliding windows are allowed to go off-bounds if they start within the left padding or the input. Sliding windows that would start in the right padded region are ignored.

Convolutional neural network example

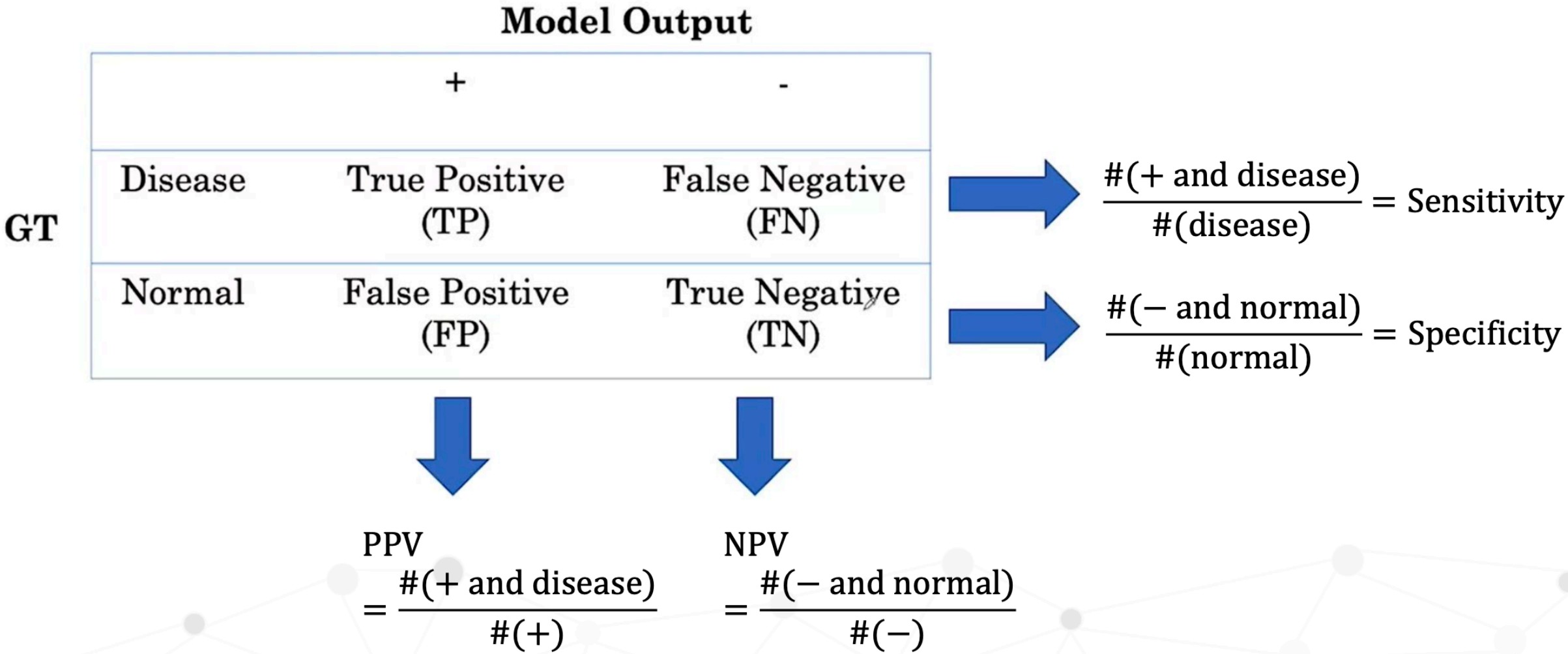
Neural network example (~LeNet-5)





Key evaluation metrics

Confusion matrix



Final Presentation

Administrative

- Midterm Review on 11/20 and Midterm on 11/27
- Research Presentation Topic
  - Presentation Week (12/18, 12/25, 1/8)
  - Volunteers of the first week and randomly pick the presentation order on 11/20
  - Pick a research paper published after 2021
  - Send the title and the paper file to TA ASAP
  - Get the confirmation from the lecturer **due on 12/4**

Project Presentation (20 mins plus 3-min Q&A)

- Abstract
  - What is this research about?
- Motivation / Purpose
  - Why do the authors perform this research?
- Introduction to the Bio-medical Imaging Technology
- Network Architecture
  - Introduce the deep learning model including its design, pros and cons
- Comparison
  - Compare the performance of this research to other researches or baselines
- Conclusion
  - Your conclusion
- Code Implementation (Bonus)
  - Show your results after running the codes (Submit the files or the link to TA)
  - Modifications are strongly encouraged!

When seeking research papers for your presentation, consider those published in renowned journals or international conferences, such as Nature, IEEE journals, OPTICA, or conferences like CVPR, ICCV, ECCV, NeurIPS, ICML, ICLR.

https://scholar.google.com/citations?view_op=top_venues&hl=zh-TW

	文章	H5 指數	H5 中位數
1.	Nature	467	707
2.	The New England Journal of Medicine	439	876
3.	Science	424	665
4.	IEEE/CVF Conference on Computer Vision and Pattern Recognition	422	681
5.	The Lancet	368	688
6.	Nature Communications	349	456
7.	Advanced Materials	326	415
8.	Cell	316	503
9.	Neural Information Processing Systems	309	503
10.	International Conference on Learning Representations	303	563
11.	JAMA	286	476
12.	Science of The Total Environment	273	375
13.	Nature Medicine	268	459
14.	Proceedings of the National Academy of Sciences	268	394
15.	Angewandte Chemie International Edition	266	362
16.	Chemical Reviews	264	459

類別 > Engineering & Computer Science > Computer Vision & Pattern Recognition

	文章	H5 指數	H5 中位數
1.	IEEE/CVF Conference on Computer Vision and Pattern Recognition	422	681
2.	European Conference on Computer Vision	238	390
3.	IEEE/CVF International Conference on Computer Vision	228	366
4.	IEEE Transactions on Pattern Analysis and Machine Intelligence	179	318
5.	IEEE Transactions on Image Processing	138	199
6.	Pattern Recognition	111	160
7.	IEEE/CVF Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)	108	176
8.	Medical Image Analysis	103	171
9.	IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)	95	150
10.	International Journal of Computer Vision	88	165
11.	Pattern Recognition Letters	80	138
12.	British Machine Vision Conference (BMVC)	77	128
13.	IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)	66	102
14.	IEEE International Conference on Image Processing (ICIP)	61	90
15.	Asian Conference on Computer Vision (ACCV)	60	92

Example

FULLY UNSUPERVISED DIVERSITY DENOISING WITH CONVOLUTIONAL VARIATIONAL AUTOENCODERS

Mangal Prakash *
Center for Systems Biology Dresden
Max-Planck Institute (CBG)
Dresden, Germany
prakash@mpi-cbg.de

Alexander Krull *†
School of Computer Science
University of Birmingham
Birmingham, UK
a.f.f.krull@bham.ac.uk

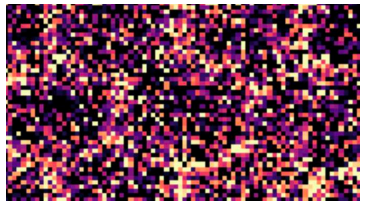
Florian Jug†
Center for Systems Biology Dresden
Max-Planck Institute (CBG)
Dresden, Germany
Fondazione Human Technopole, Milano, Italy
jug@mpi-cbg.de, florian.jug@fht.org

ABSTRACT

Deep Learning based methods have emerged as the indisputable leaders for virtually all image restoration tasks. Especially in the domain of microscopy images, various content-aware image restoration (CARE) approaches are now used to improve the interpretability of acquired data. Naturally, there are limitations to what can be restored in corrupted images, and like for all inverse problems, many potential solutions exist, and one of them must be chosen. Here, we propose DIVNOISING, a denoising approach based on fully convolutional variational autoencoders (VAEs), overcoming the problem of having to choose a single solution by predicting a whole distribution of denoised images. First we introduce a principled way of formulating the unsupervised denoising problem within the VAE framework by explicitly incorporating imaging noise models into the decoder. Our approach is fully unsupervised, only requiring noisy images and a suitable description of the imaging noise distribution. We show that such a noise model can either be measured, bootstrapped from noisy data, or co-learned during training. If desired, consensus predictions can be inferred from a set of DIVNOISING predictions, leading to competitive results with other unsupervised methods and, on occasion, even with the supervised state-of-the-art. DIVNOISING samples from the posterior enable a plethora of useful applications. We are (i) showing denoising results for

Introduction
Image Restoration

- The goal of scientific image analysis is to analyze pixel-data and measure the properties of objects of interest in images.
- Image restoration is the task of removing unwanted noise and distortions, giving us clean images that are closer to the true but unknown signal.



1. Begin by providing an overview of the background and existing research in the paper's domain.

Introduction
Variational Autoencoder (VAE)

- VAEs are generative models, capable of learning complex distributions over images.

Relationships

- Prior $p_\theta(\mathbf{z})$
- Likelihood $p_\theta(\mathbf{x}|\mathbf{z})$
- Posterior $p_\theta(\mathbf{z}|\mathbf{x})$

Instead of mapping the input into a *fixed* vector, VAE wants to map it into a distribution.

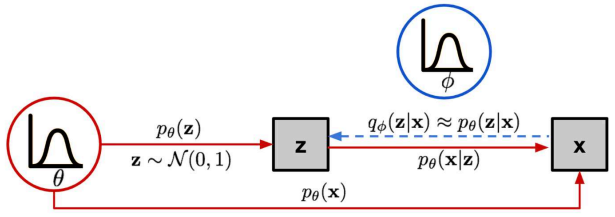


Fig. 6. The graphical model involved in Variational Autoencoder. Solid lines denote the generative distribution $p\_\theta(.)$ and dashed lines denote the distribution $q\_\phi(\mathbf{z}|\mathbf{x})$ to approximate the intractable posterior $p\_\theta(\mathbf{z}|\mathbf{x})$.

source

Introduction
Motivations

- Distortions degrade some of the information content in images, generally making it impossible to fully recover the desired clean signal with certainty.
- Generative models, such as VAEs, have been overlooked as a method to solve unsupervised image denoising problems.
  - This might also be due to the fact that vanilla VAEs show sub-par performance on denoising problems.
  - VAEs are a canonical choice when a distribution over a set of variables needs to be learned.

2. Explain the challenges or problems the authors aim to address. Highlight the key contributions of the paper.

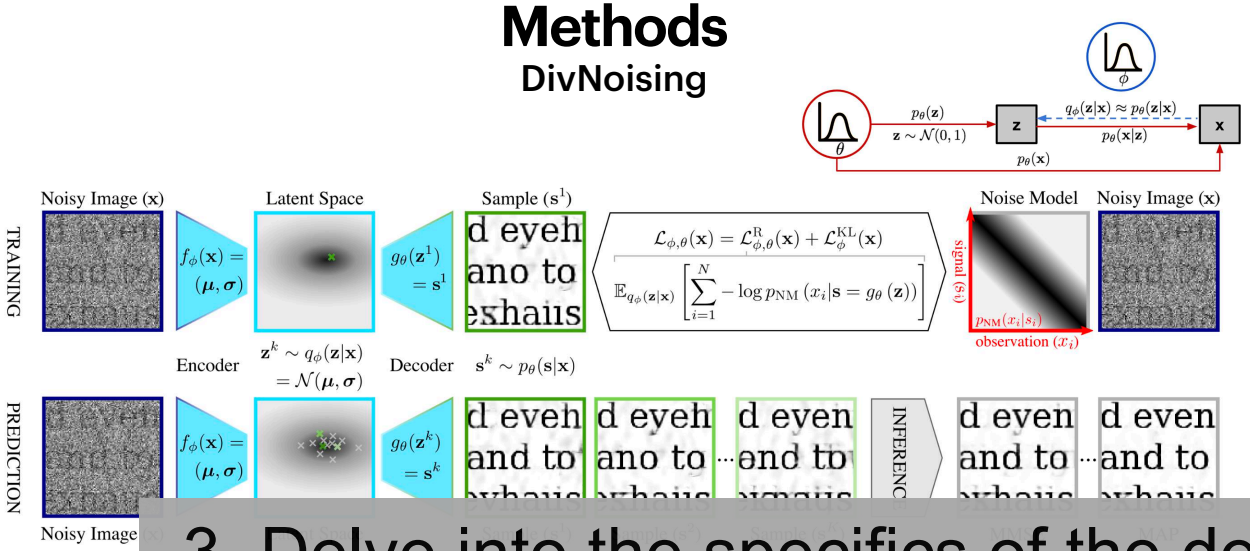
Introduction
Motivations

- Distortions degrade some of the information content in images, generally making it impossible to fully recover the desired clean signal with certainty.
- Generative models, such as VAEs, have been overlooked as a method to solve unsupervised image denoising problems.
  - This might also be due to the fact that vanilla VAEs show sub-par performance on denoising problems.
  - VAEs are a canonical choice when a distribution over a set of variables needs to be learned.

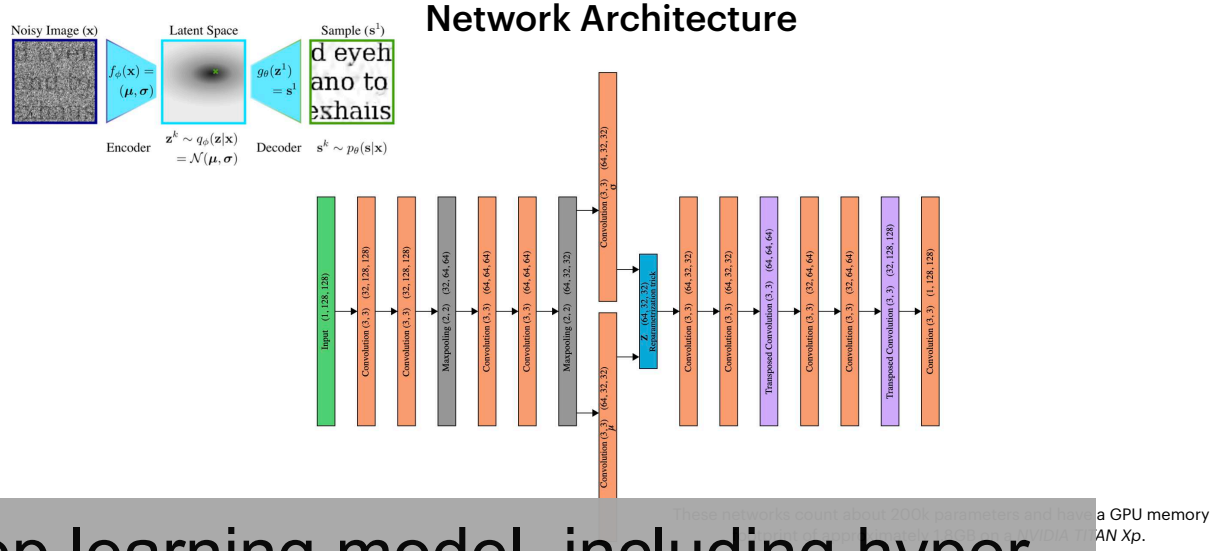
Introduction
Contributions

- DivNoisig will be hugely beneficial for computational biology applications in biomedical imaging, where noise is typically unavoidable and huge datasets need to be processed on a daily basis.
- Enable unsupervised diverse SOTA denoising while requiring only comparatively little computational resources, rendering our approach particularly practical.
- Have the potential to be useful for many real-world applications and will not only generate state-of-the-art (SOTA) restored images, but also enrich quantitative downstream processing.

Methods
DivNoising

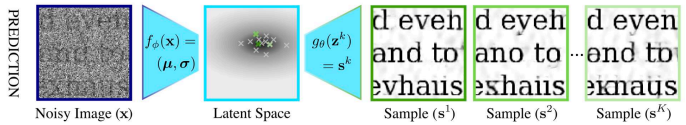


Methods
Network Architecture

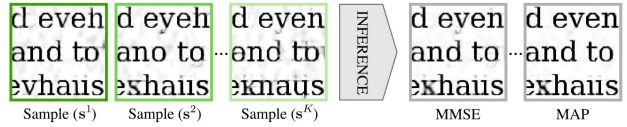


3. Delve into the specifics of the deep learning model, including hyper-parameters and loss functions. Discuss the fundamental theories or unique designs the authors used to tackle their challenges.

Prediction



Inference



- Get samples from an approximate posterior
  - 1. Feed the noisy image into our encoder.
  - 2. Draw samples $\mathbf{z}^k \sim q_\phi(\mathbf{z}|\mathbf{x})$
  - 3. Decode the samples via the decoder to get $\mathbf{s}^k = g_\theta(\mathbf{z}^k)$

- Given a set of posterior samples for a noisy image, we can infer different point estimates.
  - 1. Approximate the MMSE estimate by averaging many samples.
  - 2. Find the *maximum a posteriori* (MAP) estimate.
    - *i.e.* the most likely signal given the noisy observation, by finding the mode of the posterior distribution. For this purpose, we iteratively use the mean shift algorithm with decreasing bandwidth to find the mode of our sample set.

4. Describe the dataset used, including the number of images, the imaging technology, and whether the dataset is private or public.

Results
Dataset

- **Intrinsically Noisy data**: No priori noise model is known for microscopy datasets.
- For these datasets, we used GMM-based noise models, which are measured from calibration images, as well as co-learned noise models.
  - The *Convallaria* and mouse actin datasets are acquired on a **spinning disc confocal microscope** while the mouse skull nuclei dataset is acquired with a **point scanning confocal microscope**.
  - Widefield2SIM (W2S), acquired using a **conventional fluorescence widefield and SIM imaging**. W2S includes 144,000 real fluorescence microscopy images, resulting in a total of 360 sets of images.

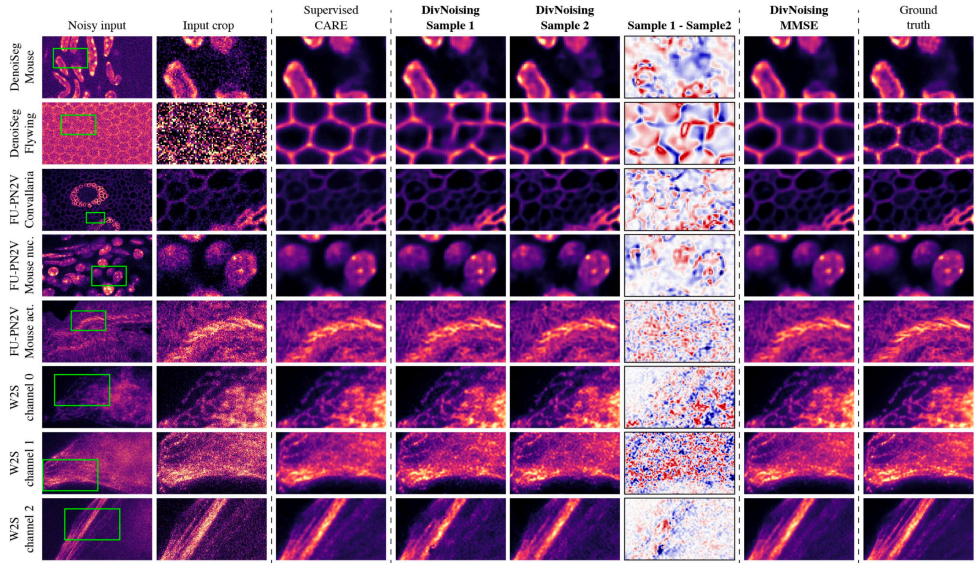
36

Results
Dataset

- **Synthesized noise dataset**
  - *DenoiSeg Mouse* data, showing cell nuclei in the developing mouse skull, consists of 908 training and 160 validation images of size 128x128, with additional 67 images of size 256x256 for testing.
    - Two noisy datasets were created with this data.
    - Exposing all images to pixel-wise independent Gaussian noise
    - First, apply poisson noise and followed by adding gaussian noise (randomly change 3% of pixels to either 0 or 255)
  - *DenoiSeg Flywing* data is showing membrane labeled cells in a fly wing, consisting of 1428 training and 252 validation patches of size 128x128, with additional 42 images of size 512x512 for testing.

37

Results

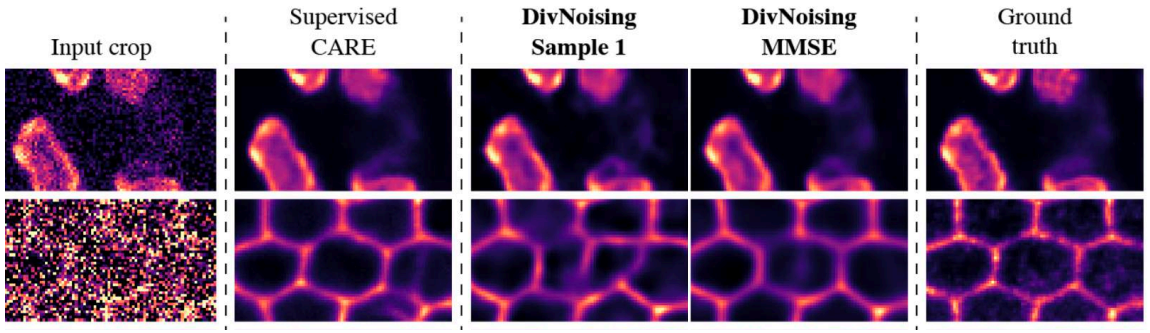


38

5. Present both quantitative and qualitative results. Clarify the evaluation metrics used and explain the significance of the images and data presented.

Results

Qualitative - Synthesized Noise Dataset



39

Results

Quantitative

		Fully Unsupervised			Unsup. (p_{NM} requ.)		Supervised
	Dataset	N2V	Vanilla VAE	DivNoising	PN2V	DivNoising	CARE
FU-PN2V	Convallaria	35.73±0.037	36.57±0.033	*36.78±0.007*	36.47±0.031	**36.90±0.004**	36.71±0.026
	↳ Bootstrapped				**36.70±0.012**	36.64±0.023	
	Mouse Act.	33.39±0.014	33.46±0.158	*33.82±0.006*	33.86±0.018	**33.99±0.004**	34.20±0.021
	Mouse Nuc.	35.84±0.015	35.84±0.023	*36.05±0.052*	**36.35±0.018**	36.26±0.047	36.58±0.019
W2S	Ch.0 (avg1)	***34.59±0.041***	33.02±0.147	34.24±0.006	-	34.13±0.002	35.22±0.069
	Ch.1 (avg1)	32.11±0.030	31.36±0.041	***32.22±0.021***	-	**32.22±0.013**	32.88±0.021
	Ch.2 (avg1)	35.04±0.073	33.72±0.187	***35.24±0.028***	32.79±0.085	35.18±0.020	35.91±0.030
	Ch.0 (avg16)	39.01±0.019	39.27±0.192	*39.45±0.036*	39.36±0.103	**39.63±0.007**	42.35±0.012
	Ch.1 (avg16)	37.91±0.059	38.33±0.021	*38.41±0.018*	**38.46±0.012**	38.39±0.007	39.64±0.061
	Ch.2 (avg16)	40.30±0.023	40.24±0.043	***40.56±0.019***	40.36±0.091	40.41±0.041	42.03±0.027
DenoiSeg	Mouse	33.84±0.070	*34.06±0.003*	*34.06±0.005*	**34.19±0.037**	34.13±0.003	35.11±0.016
	Flywing	24.79±0.034	24.88±0.045	*24.92±0.016*	24.85±0.036	**25.02±0.024**	25.79±0.014
	Mouse s&p	32.98±0.020	23.62±0.084	*35.19±0.030*	29.67±0.079	**36.21±0.015**	37.03±0.016
	BioID Face	32.34±0.080	32.58±0.022	*33.02±0.020*	**33.76±0.079**	33.12±0.039	35.06±0.051

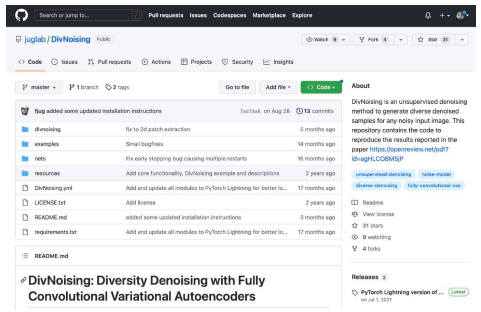
43

6. Offer your perspective on the paper. Assess the feasibility and effectiveness of the proposed methods.

Code Implementation (optional): If the authors have provided open-source code, consider trying it out or incorporating their ideas into your creative report!

Code Implementation
Overview

- Github: https://github.com/juglab/DivNoising
- I follow the official instructions to run ipython notebooks.
  - 1. Training
  - 2. Testing



Code Implementation
Testing

- Load data to predict.
- Use trained model to generate samples.
  - MMSE samples: 10 (original setting is 1000.)
- And compute PSNR.

	Fully Unsupervised			Unsup. (p_{NM} requ.)	
Dataset	N2V	Vanilla VAE	DivNoising	PN2V	DivNoising
Convallaria	35.73±0.037	36.57±0.033	*36.78±0.007*	36.47±0.031	**36.90±0.004**
↳ Bootstrapped				**36.70±0.012**	36.64±0.023

```
image: 198 psnr:32.877   mean psnr:32.818
image: 199 psnr:32.829   mean psnr:32.818
mean 32.81795153617859
```