

Training Neural Networks

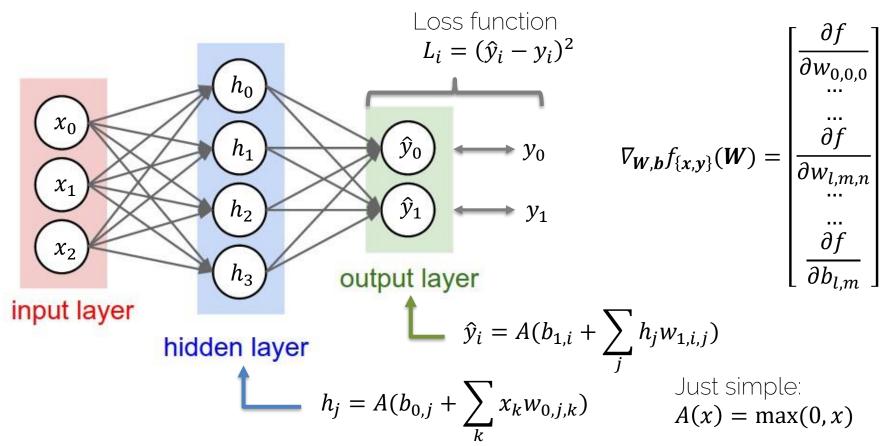
I2DL: Prof. Niessner, Prof. Leal-Taixé



Lecture 5 Recap

I2DL: Prof. Niessner, Prof. Leal-Taixé

Gradient Descent for Neural Networks



Stochastic Gradient Descent (SGD)

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^k, \boldsymbol{x}_{\{1..m\}}, \boldsymbol{y}_{\{1..m\}})$$

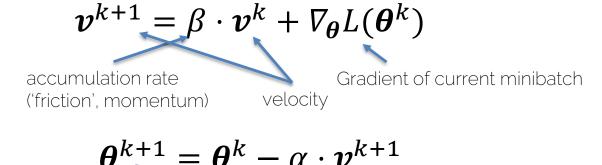
k now refers to k-th iteration

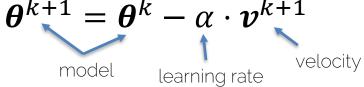
$$\nabla_{\boldsymbol{\theta}} L = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L_i$$

 $\sim m$ training samples in the current minibatch

Gradient for the *k*-th minibatch

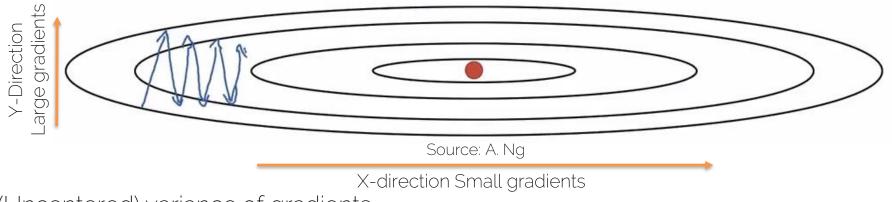
Gradient Descent with Momentum





Exponentially-weighted average of gradient Important: velocity \boldsymbol{v}^{k} is vector-valued!

RMSProp



(Uncentered) variance of gradients → second momentum

We're dividing by square gradients: - Division in Y-Direction will be large - Division in X-Direction will be

$$\boldsymbol{s}^{k+1} = \boldsymbol{\beta} \cdot \boldsymbol{s}^k + (1 - \boldsymbol{\beta}) [\nabla_{\boldsymbol{\theta}} L \circ \nabla_{\boldsymbol{\theta}} L]$$

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \cdot \frac{\nabla_{\boldsymbol{\theta}} L}{\sqrt{\boldsymbol{s}^{k+1}} + \epsilon}$$

Can increase learning rate!

small

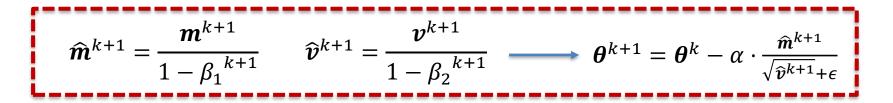
Adam

Combines Momentum and RMSProp

 $\boldsymbol{m}^{k+1} = \beta_1 \cdot \boldsymbol{m}^k + (1 - \beta_1) \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^k) \qquad \boldsymbol{v}^{k+1} = \beta_2 \cdot \boldsymbol{v}^k + (1 - \beta_2) [\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^k) \circ \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^k)]$

m^{k+1} and *v^{k+1}* are initialized with zero

 → bias towards zero
 → Typically, bias-corrected moment updates





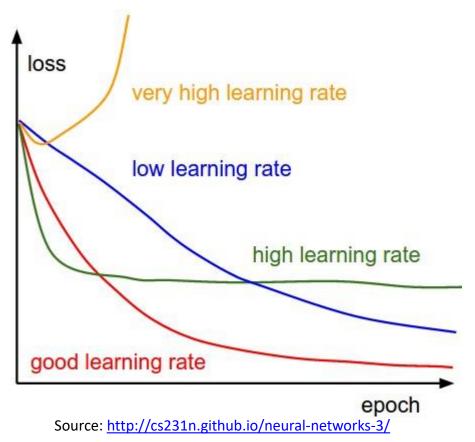
Training Neural Nets

I2DL: Prof. Niessner, Prof. Leal-Taixé

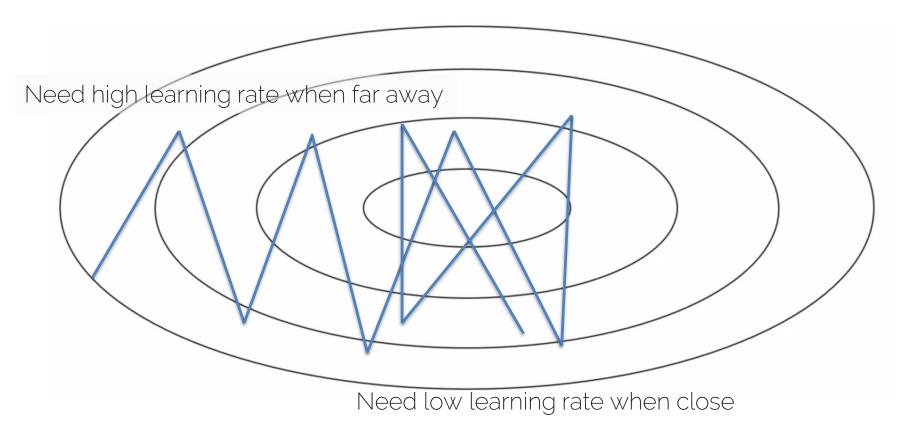
Learning Rate: Implications

• What if too high?

• What if too low?



Learning Rate



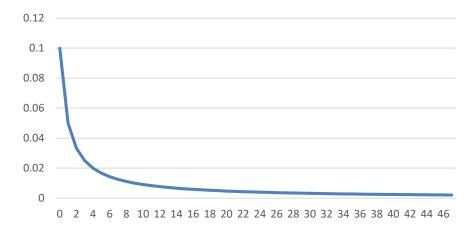
Learning Rate Decay

•
$$\alpha = \frac{1}{1 + decay_rate * epoch} \cdot \alpha_0$$

- E.g.,
$$\alpha_0 = 0.1$$
, $decay_rate = 1.0$



- → Epoch 1: 0.05
- → Epoch 2: 0.033
- → Epoch 3: 0.025



Learning Rate over Epochs

...

Learning Rate Decay

Many options:

• Step decay $\alpha = \alpha - t \cdot \alpha$ (only every n steps)

- T is decay rate (often 0.5)

- Exponential decay $\alpha = t^{epoch} \cdot \alpha_0$
 - t is decay rate (t < 1.0)

•
$$\alpha = \frac{t}{\sqrt{epoch}} \cdot a_0$$

- t is decay rate
- Etc.

Training Schedule

Manually specify learning rate for entire training process

- Manually set learning rate every n-epochs
- How?
 - Trial and error (the hard way)
 - Some experience (only generalizes to some degree)

Consider: #epochs, training set size, network size, etc.

Basic Recipe for Training

- Given ground dataset with ground labels
 - $\{x_i, y_i\}$
 - x_i is the i^{th} training image, with label y_i
 - Often $dim(x) \gg dim(y)$ (e.g., for classification)
 - *i* is often in the 100-thousands or millions
 - Take network *f* and its parameters *w*, *b*
 - Use SGD (or variation) to find optimal parameters **w**, **b**
 - Gradients from backpropagation

Gradient Descent on Train Set

- Given large train set with (n) training samples $\{x_i, y_i\}$
 - Let's say 1 million labeled images
 - Let's say our network has 500k parameters

- Gradient has 500k dimensions
- n = 1 million
- Extremely expensive to compute

Learning

- Learning means generalization to unknown dataset
 - (So far no 'real' learning)
 - I.e., train on known dataset → test with optimized parameters on unknown dataset

• Basically, we hope that based on the train set, the optimized parameters will give similar results on different data (i.e., test data)

Learning

- Training set ('*train*'):
 - Use for training your neural network
- Validation set ('*val*'):
 - Hyperparameter optimization
 - Check generalization progress
- Test set ('*test*'):
 - Only for the very end
 - NEVER TOUCH DURING DEVELOPMENT OR TRAINING

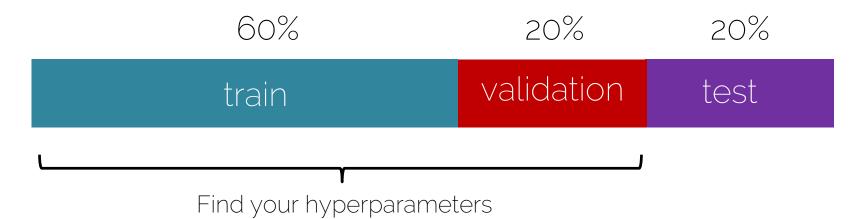
Learning

- Typical splits
 - Train (60%), Val (20%), Test (20%)
 - Train (80%), Val (10%), Test (10%)

- During training:
 - Train error comes from average minibatch error
 - Typically take subset of validation every n iterations

Basic Recipe for Machine Learning

• Split your data

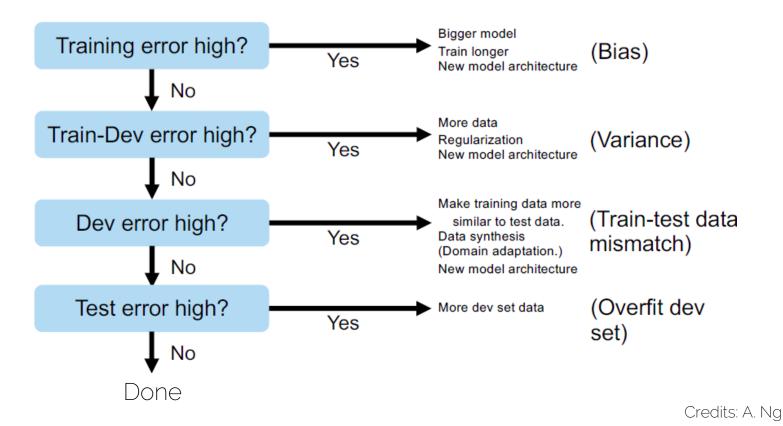


Basic Recipe for Machine Learning

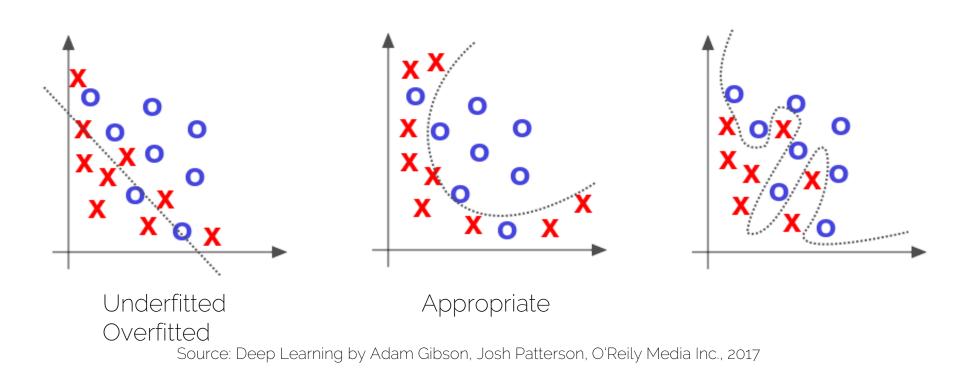
• Split your data

60%	20%	20%
train	validation	test
Ground truth error 1% Training set error 5% Val/test set error 8%	í Var	s derfitting) <i>fiance</i> erfitting)

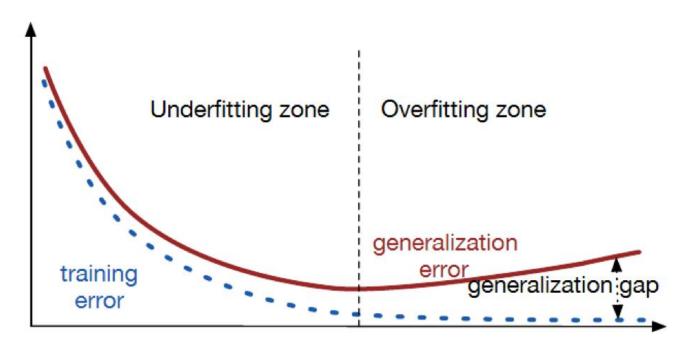
Basic Recipe for Machine Learning



Over- and Underfitting



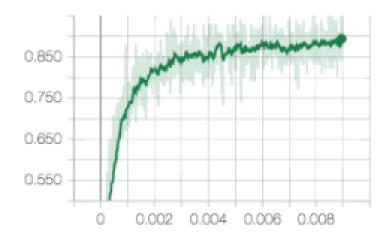
Over- and Underfitting



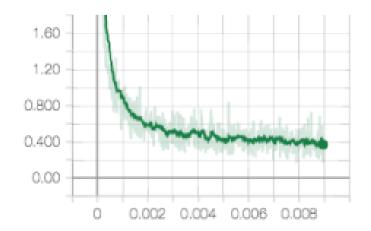
Source: https://srdas.github.io/DLBook/ImprovingModelGeneralization.html

Learning Curves

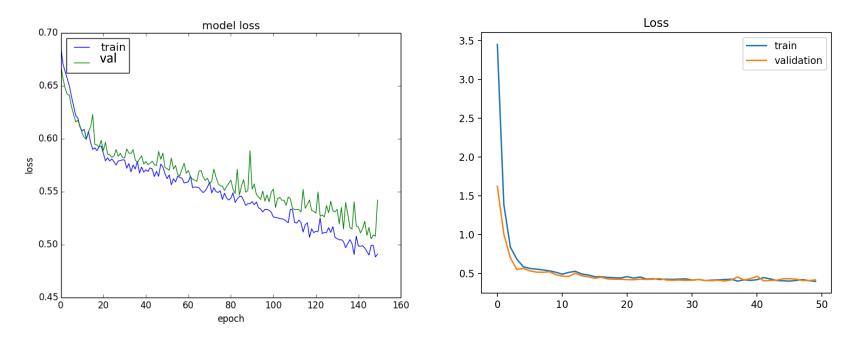
• Training graphs - Accuracy



- Loss

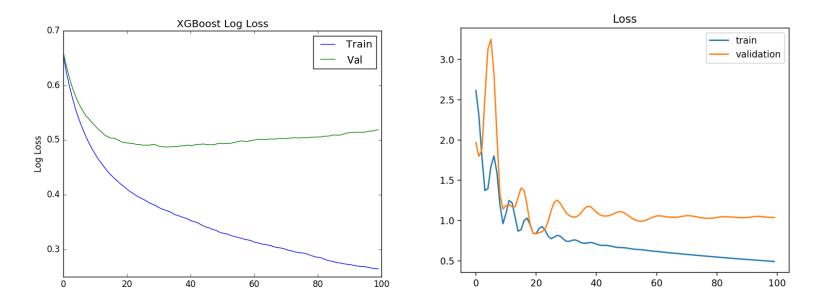


Learning Curves



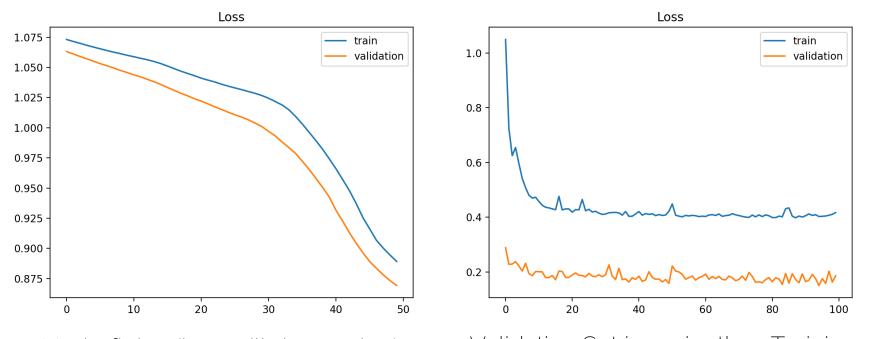
Source: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

Overfitting Curves



Source: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

Other Curves



Underfitting (loss still decreasing) Validation Set is easier than Training set Source: <u>https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/</u>

To Summarize

- Underfitting
 - Training and validation losses decrease even at the end of training
- Overfitting
 - Training loss decreases and validation loss increases
- Ideal Training
 - Small gap between training and validation loss, and both go down at same rate (stable without fluctuations).

To Summarize

- Bad Signs
 - Training error not going down
 - Validation error not going down
 - Performance on validation better than on training set
 - Tests on train set different than during training
- Bad Practice
 - Training set contains test data
 - Debug algorithm on test data

Never touch during development or training

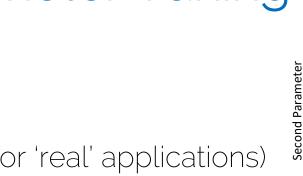
Hyperparameters

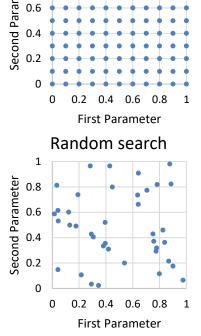
- Network architecture (e.g., num layers, #weights)
- Number of iterations
- Learning rate(s) (i.e., solver parameters, decay, etc.)
- Regularization (more later next lecture)
- Batch size
- •
- Overall: learning setup + optimization = hyperparameters

Hyperparameter Tuning

- Methods:
 - Manual search:
 - most common 🕲
 - Grid search (structured, for 'real' applications)
 - Define ranges for all parameters spaces and select points
 - Usually pseudo-uniformly distributed
 - \rightarrow Iterate over all possible configurations
 - Random search:

Like grid search but one picks points at random in the predefined ranges

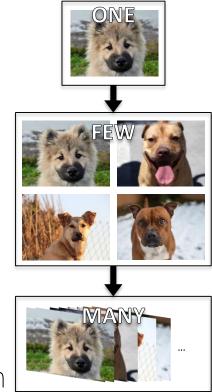




Grid search

How to Start

- Start with single training sample
 - Check if output correct
 - Overfit → train accuracy should be 100% because input just memorized
- Increase to handful of samples (e.g., **4**)
 - Check if input is handled correctly
- Move from overfitting to more samples
 - 5, 10, 100, 1000, ...
 - At some point, you should see generalization



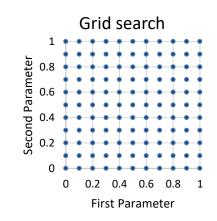
Find a Good Learning Rate

- Use all training data with small weight decay
- Perform initial loss sanity check e.g., log(C) for softmax with C classes
- Find a learning rate that makes the loss drop significantly (exponentially) within 100 iterations
- Good learning rates to try: 1e-1, 1e-2, 1e-3, 1e-4



Coarse Grid Search

- Choose a few values of learning rate and weight
 decay around what worked from
- Train a few models for a few epochs.
- Good weight decay to try: 1e-4, 1e-5, 0



Refine Grid

- Pick best models found with coarse grid.
- Refine grid search around these models.
- Train them for longer (10-20 epochs) without learning rate decay
- Study loss curves <- most important debugging tool!

Timings

- How long does each iteration take?
 - Get precise timings!
 - If an iteration exceeds 500ms, things get dicey
- Look for bottlenecks
 - Dataloading: smaller resolution, compression, train from SSD
 - Backprop
- Estimate total time
 - How long until you see some pattern? **FOR MY NEURAL NETWORK TO**
 - How long till convergence?



Network Architecture

- Frequent mistake: "Let's use this super big network, train for two weeks and we see where we stand."
- Instead: start with simplest network possible
 - Rule of thumb divide #layers you started with by 5
- Get debug cycles down
 - Ideally, minutes



Debugging

- Use train/validation/test curves
 - Evaluation needs to be consistent
 - Numbers need to be comparable

- Only make one change at a time
 - "I've added 5 more layers and double the training size, and now I also trained 5 days longer. Now it's better, but why?"

Common Mistakes in Practice

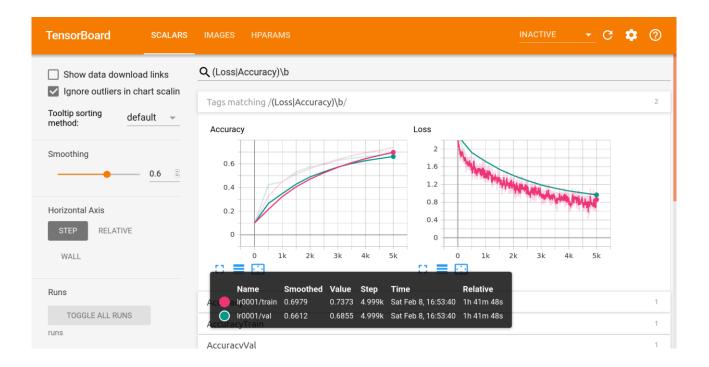
- Did not overfit to single batch first
- Forgot to toggle train/eval mode for network
 Check later when we talk about dropout...
- Forgot to call .zero_grad() (in PyTorch) before calling .backward()
- Passed softmaxed outputs to a loss function that expects raw logits



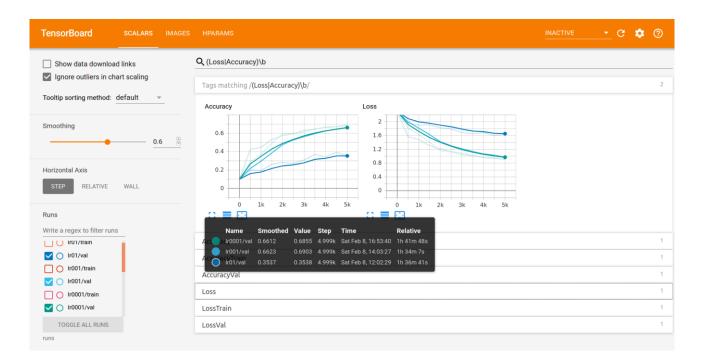
Tensorboard: Visualization in Practice

I2DL: Prof. Niessner, Prof. Leal-Taixé

Tensorboard: Compare Train/Val Curves



Tensorboard: Compare Different Runs



Tensorboard: Visualize Model Predictions

TensorBoard		IMAGES				G	٠	0
Show actual image	size		Misclassifications					9
Brightness adjustment		RESET	Misclassifications/car 1503 step 0 Sun Feb 09 2020 06:56:04 GMT+0100 (Central European Standard Time)	Misclassifications/cat 1003 step 0 Sun Feb 09 2020 06:56:04 GMT+0100 (Central European Standard Time)	Misclassifications/deer step 0 Sun Feb 09 2020 06:56:04 GMT+0100 (Central European S Time)	Itandard	N.	
Contrast adjustment		RESET				1	e	
Runs Write a regex to filter runs Image:	3		Misclassifications/dog step 0 Sun Feb 09 2020 06:56:04 GMT+0100 (Central European Standard Time) Control Control C	Misclassifications/frog teo 0 step 0 Sun Feb 09 2020 06:56:04 GMT+0100 (Central European Standard Time)	Misclassifications/horse step 0 Sun Feb 09 2020 06:56:04 GMT+0100 (Central European S Time)	itandard		
ir0001/val ir0001/val ir0001/val ir0001/S811868 ir001 ir001/S8118804 40108936 ir001 ir001/S8118804 ir01/S8118004 ir01/S8109200 ir01 ir01	2.		Misclassifications/plane step 0 Sun Feb 09 2020 06:55:04 GMT+0100 (Central European Standard Time)	Misclassifications/ship atep 0 Sun Feb 9 2020 06:56:04 GMT+01100 (Central European Standard Time)	Misclassifications/truck arte 0 Sun Feb 09 2020 06:56:04 GMT+0100 (Central European S Time)	Etandard		

Tensorboard: Visualize Model Predictions

TensorBoard SCALARS	IMAGES		0	
Show actual image size		Q Filter tags (regular expressions supported)		
Brightness adjustment		Confusion_Matrix	2	
RE	ESET	Confusion_Matrix Confus		
Contrast adjustment	ESET	pione 0.52 0.210 0.000 0.010 <th0< td=""><td></td><td></td></th0<>		
Runs Ir00		Big Big Big Big Big Big Big Big Big Big		
Ir001/train Ir001/val Ir001/train Ir001/train		transe 6 018 008 028 058 048 068 017 058 0090 03 02 transe 6 019 008 012 059 054 010 009 738 0008 012 02 ship 6 018 008 028 018 012 009 009 009 018 018 019 03 ship 6 018 008 012 059 009 019 019 009 012 059 019 019 02 ship 6 018 008 028 018 018 018 009 009 009 019 019 009 019 009 019 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 009 019 019 019 019 009 019 019 019 019 019 009 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019 019		
C Ir0001/val				
Ir0001/1581186844.0 108936		Misclassifications	19	
🔽 🔿 Ir001		1913C1023111C0110113		
O Ir001/1581188042.41 37468				
TOGGLE ALL RUNS				
runs				

Tensorboard: Compare Hyperparameters

Hyperparameters experiment_name													
 batch_size learning_rate num_epochs 		Session Group Name.	Show Metrics	experiment_name	batch_size	learning_ra	te	num_epochs	LossTrain	AccuracyTrain	L	LossVal	AccuracyVal
		Ir0001/158118684		lr0001	100.00	0.0010000		10.000	0.63350	0.77958	C	0.90479	0.68550
		Ir001/1581188042		lr001	100.00	0.010000		10.000	0.76434	0.73432	C	0.90190	0.69030
Metrics LossTrain	_	Ir01/1581192020		lr01	100.00	0.10000		10.000	1.6232	0.36526	1	1.6357	0.35380
Min -infinity	Max +infinity												
AccuracyTrain													
Min	Max +infinity												
-infinity	+infinity												
							Color by	-					
							AccuracyVal	·	AccuracyTrain		LossVal		AccuracyVal
<u> </u>	AccuracyTrain	LossVal		Accurac			AccuracyVal	mic	 Linear Logarithmic 		 Linear Logarithm 	nic	 Linear Logarithmic
	Linear Logarithmic	 Linear Logarithm 	ic	O Line O Log	ar arithmic		AccuracyVal	mic	Linear		Linear	nie	 Linear
÷.	 Linear 	Linear	ic	O Line	ar arithmic		AccuracyVal	mic	 Linear Logarithmic 		 Linear Logarithm 	nic	 Linear Logarithmic
	Linear Logarithmic Quantile	 Linear Logarithm 	lc AccuracyTrain	O Line O Log O Qua	ar arithmic ntile		AccuracyVal	mic	 Linear Logarithmic 		 Linear Logarithm 	nic	 Linear Logarithmic
	Linear Logarithmic Quantile	Linear Logarithm Quantile	AccuracyTrain 0.75	O Line O Log @ Qua	ar arithmic ntile		AccuracyVal LosaTrain © Linear Logarith Quantile Quantile	mic	 Linear Logarithmic 		 Linear Logarithm 	nic	 Linear Logarithmic
experiment_nam	Linear Logarithmic Quantile	Linear Logarithm Quantile	AccuracyTrain	O Line O Log @ Qua	ar arithmic ntile		AccuracyVal LosaTrein LosaTrein Linear Quantile AccuracyVal Quantile AccuracyVal Accuracy	mic	 Linear Logarithmic 		 Linear Logarithm 	nic	 Linear Logarithmic
experiment_nam	Linear Logarithmic Quantile	Linear Logarithm Quantile	AccuracyTrain 0.75 ² 0.77 65 - 0.60 -	O Line O Log @ Qua	ar arithmic ntile		AccuracyVal LosaTrain © Linear O Logarith Quantile Quantile 0 0 0 0 0 0 0 0 0 0 0 0 0	mic	 Linear Logarithmic 		 Linear Logarithm 	nic	 Linear Logarithmic
experiment nam	Linear Logarithmic Quantile	Cogarithm Cogarithm Quantile LossTan LossTan	AccuracyTrain 0.75- 0.75- 0.75-	O Line O Log @ Qua	ar arithmic ntile		AccuracyVal LossTrain Ucosertith Quantile Quantile Accord	mic	 Linear Logarithmic 		 Linear Logarithm 		 Linear Logarithmic
experiment_nam	Logarithmic Quantile setting.ste a	Cogarithm Cogarithm Cogarithm Quantile	AccuracyTrain 0.75 0.55 0.60 0.55	O Line O Log @ Qua	ar arithmic ntile		AccuracyVal	mic	 Linear Logarithmic 		 Linear Logarithm 		 Linear Logarithmic
experiment nam	Linear Logarithmic Quantile	Cogarithm Cogarithm Cogarithm Quantile	Accuracy Brain 0.75 0.57 0.60 0.60 0.55 0.50 0.55 0.50 0.55	Lossival Accurac 0 45000 16- 15- 14- 13- 0 45000 0 450000 0 45000 0 450000 0 450000 0 45000 0 4500000000000000000000000000000000000	ar arithmic ntile		AccuracyVal LosaTrain © Logarith O Logarith O Quantile Moreever 0 4 16 16 16 16 16 16 16 16 16 16	mic	 Linear Logarithmic 		 Linear Logarithm 		 Linear Logarithmic

Next Lecture

- Next lecture
 - More about training neural networks: output functions, loss functions, activation functions

• Check the exercises 🕲



See you next week 🕲

I2DL: Prof. Niessner, Prof. Leal-Taixé

References

- Goodfellow et al. "Deep Learning" (2016),
 Chapter 6: Deep Feedforward Networks
- Bishop "Pattern Recognition and Machine Learning" (2006),
 Chapter 5.5: Regularization in Network Nets
- <u>http://cs231n.github.io/neural-networks-1/</u>
- <u>http://cs231n.github.io/neural-networks-2/</u>
- <u>http://cs231n.github.io/neural-networks-3/</u>