

The Incredible PYTORCH

This is a curated list of tutorials, projects, libraries, videos, papers, books and anything related to the incredible [PyTorch](#). Feel free to make a pull request to contribute to this list.

Tutorials

- [Official PyTorch Tutorials](#)
- [Official PyTorch Examples](#)
- [Practical Deep Learning with PyTorch](#)
- [Dive Into Deep Learning with PyTorch](#)
- [Deep Learning Models](#)
- [Minicourse in Deep Learning with PyTorch](#)
- [C++ Implementation of PyTorch Tutorial](#)
- [Simple Examples to Introduce PyTorch](#)
- [Mini Tutorials in PyTorch](#)
- [Deep Learning for NLP](#)
- [Deep Learning Tutorial for Researchers](#)
- [Fully Convolutional Networks implemented with PyTorch](#)
- [Simple PyTorch Tutorials Zero to ALL](#)
- [DeepNLP-models-Pytorch](#)
- [MILA PyTorch Welcome Tutorials](#)
- [Effective PyTorch, Optimizing Runtime with TorchScript and Numerical Stability Optimization](#)
- [Practical PyTorch](#)
- [PyTorch Project Template](#)

Visualization

- [Loss Visualization](#)
- [Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization](#)

- [Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps](#)
- [SmoothGrad: removing noise by adding noise](#)
- [DeepDream: dream-like hallucinogenic visuals](#)
- [FlashTorch: Visualization toolkit for neural networks in PyTorch](#)
- [Lucent: Lucid adapted for PyTorch](#)

Explainability

- [Efficient Covariance Estimation from Temporal Data](#)
- [Hierarchical interpretations for neural network predictions](#)
- [Shap, a unified approach to explain the output of any machine learning model](#)
- [Visualizing PyTorch saved .pth deep learning models with netron](#)
- [Distilling a Neural Network Into a Soft Decision Tree](#)

Object Detection

- [MMDetection Object Detection Toolbox](#)
- [Mask R-CNN Benchmark: Faster R-CNN and Mask R-CNN in PyTorch 1.0](#)
- [YOLOv3](#)
- [YOLOv2: Real-Time Object Detection](#)
- [SSD: Single Shot MultiBox Detector](#)
- [Detectron models for Object Detection](#)
- [Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks](#)
- [Whale Detector](#)

Long-Tailed / Out-of-Distribution Recognition

- [Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization](#)
- [Invariant Risk Minimization](#)
- [Training Confidence-Calibrated Classifier for Detecting Out-of-Distribution Samples](#)
- [Deep Anomaly Detection with Outlier Exposure](#)
- [Large-Scale Long-Tailed Recognition in an Open World](#)
- [Principled Detection of Out-of-Distribution Examples in Neural Networks](#)
- [Learning Confidence for Out-of-Distribution Detection in Neural Networks](#)

- [PyTorch Imbalanced Class Sampler](#)

Energy-Based Learning

- [EBGAN, Energy-Based GANs](#)
- [Maximum Entropy Generators for Energy-based Models](#)

Missing Data

- [BRITS: Bidirectional Recurrent Imputation for Time Series](#)

Architecture Search

- [DenseNAS](#)
- [DARTS: Differentiable Architecture Search](#)
- [Efficient Neural Architecture Search \(ENAS\)](#)
- [EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks](#)

Optimization

- [AccSGD, AdaBound, AdaMod, DiffGrad, Lamb, NovoGrad, RAdam, SGDW, Yogi and more](#)
- [Lookahead Optimizer: k steps forward, 1 step back](#)
- [RAdam, On the Variance of the Adaptive Learning Rate and Beyond](#)
- [Over9000, Comparison of RAdam, Lookahead, Novograd, and combinations](#)
- [AdaBound, Train As Fast as Adam As Good as SGD](#)
- [Riemannian Adaptive Optimization Methods](#)
- [L-BFGS](#)
- [OptNet: Differentiable Optimization as a Layer in Neural Networks](#)
- [Learning to learn by gradient descent by gradient descent](#)

Quantization

- [Additive Power-of-Two Quantization: An Efficient Non-uniform Discretization For Neural Networks](#)

Quantum Machine Learning

- [Tor10](#), generic tensor-network library for quantum simulation in PyTorch
- [PennyLane](#), cross-platform Python library for quantum machine learning with PyTorch interface

Neural Network Compression

- [Bayesian Compression for Deep Learning](#)
- [Neural Network Distiller by Intel AI Lab: a Python package for neural network compression research](#)
- [Learning Sparse Neural Networks through L0 regularization](#)
- [Energy-constrained Compression for Deep Neural Networks via Weighted Sparse Projection and Layer Input Masking](#)
- [EigenDamage: Structured Pruning in the Kronecker-Factored Eigenbasis](#)
- [Pruning Convolutional Neural Networks for Resource Efficient Inference](#)
- [Pruning neural networks: is it time to nip it in the bud? \(showing reduced networks work better\)](#)

Facial, Action and Pose Recognition

- [Facenet: Pretrained Pytorch face detection and recognition models](#)
- [DGC-Net: Dense Geometric Correspondence Network](#)
- [High performance facial recognition library on PyTorch](#)
- [FaceBoxes](#), a CPU real-time face detector with high accuracy
- [How far are we from solving the 2D & 3D Face Alignment problem? \(and a dataset of 230,000 3D facial landmarks\)](#)
- [Learning Spatio-Temporal Features with 3D Residual Networks for Action Recognition](#)
- [PyTorch Realtime Multi-Person Pose Estimation](#)
- [SphereFace: Deep Hypersphere Embedding for Face Recognition](#)
- [GANimation: Anatomically-aware Facial Animation from a Single Image](#)
- [Shufflenet V2 by Face++ with better results than paper](#)
- [Towards 3D Human Pose Estimation in the Wild: a Weakly-supervised Approach](#)
- [Unsupervised Learning of Depth and Ego-Motion from Video](#)
- [FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks](#)
- [FlowNet: Learning Optical Flow with Convolutional Networks](#)
- [Optical Flow Estimation using a Spatial Pyramid Network](#)
- [OpenFace in PyTorch](#)
- [Deep Face Recognition in PyTorch](#)

Super resolution

- [Enhanced Deep Residual Networks for Single Image Super-Resolution](#)
- [Superresolution using an efficient sub-pixel convolutional neural network](#)
- [Perceptual Losses for Real-Time Style Transfer and Super-Resolution](#)

Synthetizing Views

- [NeRF, Neural Radian Fields, Synthesizing Novels Views of Complex Scenes](#)

Voice

- [Google AI VoiceFilter: Targeted Voice Separation by Speaker-Conditioned Spectrogram Masking](#)

Medical

- [U-Net for FLAIR Abnormality Segmentation in Brain MRI](#)
- [Genomic Classification via ULMFiT](#)
- [Deep Neural Networks Improve Radiologists' Performance in Breast Cancer Screening](#)
- [Delira, lightweight framework for medical imaging prototyping](#)
- [V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation](#)
- [Medical Torch, medical imaging framework for PyTorch](#)

3D Segmentation, Classification and Regression

- [Kaolin, Library for Accelerating 3D Deep Learning Research](#)
- [PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation](#)

Video Recognition

- [Dancing to Music](#)
- [Devil Is in the Edges: Learning Semantic Boundaries from Noisy Annotations](#)
- [Deep Video Analytics](#)

- [PredRNN: Recurrent Neural Networks for Predictive Learning using Spatiotemporal LSTMs](#)

Recurrent Neural Networks (RNNs)

- [Ordered Neurons: Integrating Tree Structures into Recurrent Neural Networks](#)
- [Averaged Stochastic Gradient Descent with Weight Dropped LSTM](#)
- [Training RNNs as Fast as CNNs](#)
- [Quasi-Recurrent Neural Network \(QRNN\)](#)
- [ReSeg: A Recurrent Neural Network-based Model for Semantic Segmentation](#)
- [A Recurrent Latent Variable Model for Sequential Data \(VRNN\)](#)
- [Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks](#)
- [Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling](#)
- [Attentive Recurrent Comparators](#)
- [Collection of Sequence to Sequence Models with PyTorch](#)
 - i. Vanilla Sequence to Sequence models
 - ii. Attention based Sequence to Sequence models
 - iii. Faster attention mechanisms using dot products between the final encoder and decoder hidden states

Convolutional Neural Networks (CNNs)

- [LegoNet: Efficient Convolutional Neural Networks with Lego Filters](#)
- [MeshCNN, a convolutional neural network designed specifically for triangular meshes](#)
- [Octave Convolution](#)
- [PyTorch Image Models, ResNet/ResNeXT, DPN, MobileNet-V3/V2/V1, MNASNet, Single-Path NAS, FBNet](#)
- [Deep Neural Networks with Box Convolutions](#)
- [Invertible Residual Networks](#)
- [Stochastic Downsampling for Cost-Adjustable Inference and Improved Regularization in Convolutional Networks](#)
- [Faster Faster R-CNN Implementation](#)
 - [Faster R-CNN Another Implementation](#)
- [Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer](#)

- [Wide ResNet model in PyTorch -DiracNets: Training Very Deep Neural Networks Without Skip-Connections](#)
- [An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition](#)
- [Efficient Densenet](#)
- [Video Frame Interpolation via Adaptive Separable Convolution](#)
- [Learning local feature descriptors with triplets and shallow convolutional neural networks](#)
- [Densely Connected Convolutional Networks](#)
- [Very Deep Convolutional Networks for Large-Scale Image Recognition](#)
- [SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size](#)
- [Deep Residual Learning for Image Recognition](#)
- [Training Wide ResNets for CIFAR-10 and CIFAR-100 in PyTorch](#)
- [Deformable Convolutional Network](#)
- [Convolutional Neural Fabrics](#)
- [Deformable Convolutional Networks in PyTorch](#)
- [Dilated ResNet combination with Dilated Convolutions](#)
- [Striving for Simplicity: The All Convolutional Net](#)
- [Convolutional LSTM Network](#)
- [Big collection of pretrained classification models](#)
- [PyTorch Image Classification with Kaggle Dogs vs Cats Dataset](#)
- [CIFAR-10 on Pytorch with VGG, ResNet and DenseNet](#)
- [Base pretrained models and datasets in pytorch \(MNIST, SVHN, CIFAR10, CIFAR100, STL10, AlexNet, VGG16, VGG19, ResNet, Inception, SqueezeNet\)](#)
- [NVIDIA/unsupervised-video-interpolation](#)

Segmentation

- [Detectron2 by FAIR](#)
- [Pixel-wise Segmentation on VOC2012 Dataset using PyTorch](#)
- [Pywick - High-level batteries-included neural network training library for Pytorch](#)
- [Improving Semantic Segmentation via Video Propagation and Label Relaxation](#)

Geometric Deep Learning: Graph & Irregular Structures

- [PyTorch Geometric, Deep Learning Extension](#)
- [Self-Attention Graph Pooling](#)

- [Position-aware Graph Neural Networks](#)
- [Signed Graph Convolutional Neural Network](#)
- [Graph U-Nets](#)
- [Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks](#)
- [MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing](#)
- [Semi-Supervised Graph Classification: A Hierarchical Graph Perspective](#)
- [PyTorch BigGraph by FAIR for Generating Embeddings From Large-scale Graph Data](#)
- [Capsule Graph Neural Network](#)
- [Splitter: Learning Node Representations that Capture Multiple Social Contexts](#)
- [A Higher-Order Graph Convolutional Layer](#)
- [Predict then Propagate: Graph Neural Networks meet Personalized PageRank](#)
- [Lorentz Embeddings: Learn Continuous Hierarchies in Hyperbolic Space](#)
- [Graph Wavelet Neural Network](#)
- [Watch Your Step: Learning Node Embeddings via Graph Attention](#)
- [Signed Graph Convolutional Network](#)
- [Graph Classification Using Structural Attention](#)
- [SimGNN: A Neural Network Approach to Fast Graph Similarity Computation](#)
- [SINE: Scalable Incomplete Network Embedding](#)
- [HyperER: Hypernetwork Knowledge Graph Embeddings](#)
- [TuckER: Tensor Factorization for Knowledge Graph Completion](#)

Sorting

- [Stochastic Optimization of Sorting Networks via Continuous Relaxations](#)

Ordinary Differential Equations Networks

- [Latent ODEs for Irregularly-Sampled Time Series](#)
- [GRU-ODE-Bayes: continuous modelling of sporadically-observed time series](#)

Multi-task Learning

- [Hierarchical Multi-Task Learning Model](#)

Steve Nouri

- [Task-based End-to-end Model Learning](#)

GANs, VAEs, and AEs

- [Mimicry, PyTorch Library for Reproducibility of GAN Research](#)
- [Clean Readable CycleGAN](#)
- [StarGAN](#)
- [Block Neural Autoregressive Flow](#)
- [High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs](#)
- [A Style-Based Generator Architecture for Generative Adversarial Networks](#)
- [GANDissect, PyTorch Tool for Visualizing Neurons in GANs](#)
- [Learning deep representations by mutual information estimation and maximization](#)
- [Variational Laplace Autoencoders](#)
- [VeGANS, library for easily training GANs](#)
- [Progressive Growing of GANs for Improved Quality, Stability, and Variation](#)
- [Conditional GAN](#)
- [Wasserstein GAN](#)
- [Adversarial Generator-Encoder Network](#)
- [Image-to-Image Translation with Conditional Adversarial Networks](#)
- [Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks](#)
- [On the Effects of Batch and Weight Normalization in Generative Adversarial Networks](#)
- [Improved Training of Wasserstein GANs](#)
- [Collection of Generative Models with PyTorch](#)
 - Generative Adversarial Nets (GAN)
 - a. [Vanilla GAN](#)
 - b. [Conditional GAN](#)
 - c. [InfoGAN](#)
 - d. [Wasserstein GAN](#)
 - e. [Mode Regularized GAN](#)
 - Variational Autoencoder (VAE)
 - a. [Vanilla VAE](#)
 - b. [Conditional VAE](#)
 - c. [Denoising VAE](#)
 - d. [Adversarial Autoencoder](#)

- e. [Adversarial Variational Bayes](#)
- [Improved Training of Wasserstein GANs](#)
- [CycleGAN and Semi-Supervised GAN](#)
- [Improving Variational Auto-Encoders using Householder Flow and using convex combination linear Inverse Autoregressive Flow](#)
- [PyTorch GAN Collection](#)
- [Generative Adversarial Networks, focusing on anime face drawing](#)
- [Simple Generative Adversarial Networks](#)
- [Adversarial Auto-encoders](#)
- [torchgan: Framework for modelling Generative Adversarial Networks in Pytorch](#)

Unsupervised Learning

- [Unsupervised Embedding Learning via Invariant and Spreading Instance Feature](#)
- [AND: Anchor Neighbourhood Discovery](#)

Adversarial Attacks

- [Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images](#)
- [Explaining and Harnessing Adversarial Examples](#)
- [AdverTorch - A Toolbox for Adversarial Robustness Research](#)

Style Transfer

- [Detecting Adversarial Examples via Neural Fingerprinting](#)
- [A Neural Algorithm of Artistic Style](#)
- [Multi-style Generative Network for Real-time Transfer](#)
- [DeOldify, Coloring Old Images](#)
- [Neural Style Transfer](#)
- [Fast Neural Style Transfer](#)
- [Draw like Bob Ross](#)

Image Captioning

- [NeuralTalk 2, Image Captioning Model, in PyTorch](#)
- [Generate captions from an image with PyTorch](#)

Transformers

- [Attention is all you need](#)
- [Spatial Transformer Networks](#)

Similarity Networks and Functions

- [Conditional Similarity Networks](#)

Reasoning

- [Inferring and Executing Programs for Visual Reasoning](#)

General NLP

- [Espresso, Module Neural Automatic Speech Recognition Toolkit](#)
- [Label-aware Document Representation via Hybrid Attention for Extreme Multi-Label Text Classification](#)
- [XLNet](#)
- [Conversing by Reading: Contentful Neural Conversation with On-demand Machine Reading](#)
- [Cross-lingual Language Model Pretraining](#)
- [Libre Office Translate via PyTorch NMT](#)
- [BERT](#)
- [VSE++: Improved Visual-Semantic Embeddings](#)
- [A Structured Self-Attentive Sentence Embedding](#)
- [Neural Sequence labeling model](#)
- [Skip-Thought Vectors](#)
- [Complete Suite for Training Seq2Seq Models in PyTorch](#)
- [MUSE: Multilingual Unsupervised and Supervised Embeddings](#)

Question and Answering

- [Visual Question Answering in Pytorch](#)
- [Reading Wikipedia to Answer Open-Domain Questions](#)
- [Deal or No Deal? End-to-End Learning for Negotiation Dialogues](#)
- [Interpretable Counting for Visual Question Answering](#)

- [Open Source Chatbot with PyTorch](#)

Speech Generation and Recognition

- [PyTorch-Kaldi Speech Recognition Toolkit](#)
- [WaveGlow: A Flow-based Generative Network for Speech Synthesis](#)
- [OpenNMT](#)
- [Deep Speech 2: End-to-End Speech Recognition in English and Mandarin](#)

Document and Text Classification

- [Hierarchical Attention Network for Document Classification](#)
- [Hierarchical Attention Networks for Document Classification](#)
- [CNN Based Text Classification](#)

Text Generation

- [Pytorch Poetry Generation](#)

Translation

- [Open-source \(MIT\) Neural Machine Translation \(NMT\) System](#)

Sentiment Analysis

- [Recurrent Neural Networks for Sentiment Analysis \(Aspect-Based\) on SemEval 2014](#)
- [Seq2Seq Intent Parsing](#)
- [Finetuning BERT for Sentiment Analysis](#)

Deep Reinforcement Learning

- [Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels](#)
- [Exploration by Random Network Distillation](#)
- [EGG: Emergence of lanGuage in Games, quickly implement multi-agent games with discrete channel communication](#)
- [Temporal Difference VAE](#)

Steve Nouri

- [High-performance Atari A3C Agent in 180 Lines PyTorch](#)
- [Learning when to communicate at scale in multiagent cooperative and competitive tasks](#)
- [Actor-Attention-Critic for Multi-Agent Reinforcement Learning](#)
- [PPO in PyTorch C++](#)
- [Reinforcement Learning for Bandit Neural Machine Translation with Simulated Human Feedback](#)
- [Asynchronous Methods for Deep Reinforcement Learning](#)
- [Continuous Deep Q-Learning with Model-based Acceleration](#)
- [Asynchronous Methods for Deep Reinforcement Learning for Atari 2600](#)
- [Trust Region Policy Optimization](#)
- [Neural Combinatorial Optimization with Reinforcement Learning](#)
- [Noisy Networks for Exploration](#)
- [Distributed Proximal Policy Optimization](#)
- [Reinforcement learning models in ViZDoom environment with PyTorch](#)
- [Reinforcement learning models using Gym and Pytorch](#)
- [SLM-Lab: Modular Deep Reinforcement Learning framework in PyTorch](#)

Deep Bayesian Learning and Probabilistic Programming

- [BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning](#)
- [Subspace Inference for Bayesian Deep Learning](#)
- [Bayesian Deep Learning with Variational Inference Package](#)
- [Probabilistic Programming and Statistical Inference in PyTorch](#)
- [Bayesian CNN with Variational Inference in PyTorch](#)

Spiking Neural Networks

- [Norse, Library for Deep Learning with Spiking Neural Networks](#)

Anomaly Detection

- [Detection of Accounting Anomalies using Deep Autoencoder Neural Networks](#)

Regression Types

Steve Nouri

- [Quantile Regression DQN](#)

Time Series

- [Dual Self-Attention Network for Multivariate Time Series Forecasting](#)
- [DILATE: DIsortion Loss with shApe and tImE](#)
- [Variational Recurrent Autoencoder for Timeseries Clustering](#)
- [Spatio-Temporal Neural Networks for Space-Time Series Modeling and Relations Discovery](#)

Synthetic Datasets

- [Meta-Sim: Learning to Generate Synthetic Datasets](#)

Neural Network General Improvements

- [In-Place Activated BatchNorm for Memory-Optimized Training of DNNs](#)
- [Train longer, generalize better: closing the generalization gap in large batch training of neural networks](#)
- [FreezeOut: Accelerate Training by Progressively Freezing Layers](#)
- [Binary Stochastic Neurons](#)
- [Compact Bilinear Pooling](#)
- [Mixed Precision Training in PyTorch](#)

DNN Applications in Chemistry and Physics

- [Wave Physics as an Analog Recurrent Neural Network](#)
- [Neural Message Passing for Quantum Chemistry](#)
- [Automatic chemical design using a data-driven continuous representation of molecules](#)
- [Deep Learning for Physical Processes: Integrating Prior Scientific Knowledge](#)

New Thinking on General Neural Network Architecture

- [Complement Objective Training](#)
- [Decoupled Neural Interfaces using Synthetic Gradients](#)

Linear Algebra

- [Eigenvectors from Eigenvalues](#)

API Abstraction

- [Torch Layers, Shape inference for PyTorch, SOTA Layers](#)

Low Level Utilities

- [TorchSharp, .NET API with access to underlying library powering PyTorch](#)

PyTorch Utilities

- [PyTorch Metric Learning](#)
- [Kornia: an Open Source Differentiable Computer Vision Library for PyTorch](#)
- [BackPACK to easily Extract Variance, Diagonal of Gauss-Newton, and KFAC](#)
- [PyHessian for Computing Hessian Eigenvalues, trace of matrix, and ESD](#)
- [Hessian in PyTorch](#)
- [Differentiable Convex Layers](#)
- [Albumentations: Fast Image Augmentation Library](#)
- [Higher, obtain higher order gradients over losses spanning training loops](#)
- [Neural Pipeline, Training Pipeline for PyTorch](#)
- [Layer-by-layer PyTorch Model Profiler for Checking Model Time Consumption](#)
- [Sparse Distributions](#)
- [Diffdist, Adds Support for Differentiable Communication allowing distributed model parallelism](#)
- [HessianFlow, Library for Hessian Based Algorithms](#)
- [Texar, PyTorch Toolkit for Text Generation](#)
- [PyTorch FLOPs counter](#)
- [PyTorch Inference on C++ in Windows](#)
- [EuclidesDB, Multi-Model Machine Learning Feature Database](#)
- [Data Augmentation and Sampling for Pytorch](#)
- [PyText, deep learning based NLP modelling framework officially maintained by FAIR](#)
- [Torchstat for Statistics on PyTorch Models](#)
- [Load Audio files directly into PyTorch Tensors](#)

- [Weight Initializations](#)
- [Spatial transformer implemented in PyTorch](#)
- [PyTorch AWS AML, run PyTorch with GPU support in less than 5 minutes](#)
- [Use tensorboard with PyTorch](#)
- [Simple Fit Module in PyTorch, similar to Keras](#)
- [torchbearer: A model fitting library for PyTorch](#)
- [PyTorch to Keras model converter](#)
- [Gluon to PyTorch model converter with code generation](#)
- [Catalyst: High-level utils for PyTorch DL & RL research](#)

PyTorch Video Tutorials

- [Practical Deep Learning with PyTorch](#)
- [PyTorch Zero to All Lectures](#)

Datasets

- [Worldbank Data](#)

Community

- [PyTorch Discussion Forum](#)
- [StackOverflow PyTorch Tags](#)

Links to This Repository

- [Github Repository](#)
- [Website](#)

To be Classified

- [Perturbative Neural Networks](#)
- [Accurate Neural Network Potential](#)
- [Scaling the Scattering Transform: Deep Hybrid Networks](#)
- [CortexNet: a Generic Network Family for Robust Visual Temporal Representations](#)
- [Oriented Response Networks](#)
- [Associative Compression Networks](#)

- [Clarinet](#)
- [Continuous Wavelet Transforms](#)
- [mixup: Beyond Empirical Risk Minimization](#)
- [Network In Network](#)
- [Highway Networks](#)
- [Hybrid computing using a neural network with dynamic external memory](#)
- [Value Iteration Networks](#)
- [Differentiable Neural Computer](#)
- [A Neural Representation of Sketch Drawings](#)
- [Understanding Deep Image Representations by Inverting Them](#)
- [NIMA: Neural Image Assessment](#)
- [NASNet-A-Mobile. Ported weights](#)
- [Graphics code generating model using Processing](#)

TensorFlow Examples and Tutorials

Tutorial index

0 - Prerequisite

- [Introduction to Machine Learning](#).
- [Introduction to MNIST Dataset](#).

1 - Introduction

- **Hello World** ([notebook](#)). Very simple example to learn how to print "hello world" using TensorFlow 2.0.
- **Basic Operations** ([notebook](#)). A simple example that cover TensorFlow 2.0 basic operations.

2 - Basic Models

- **Linear Regression** ([notebook](#)). Implement a Linear Regression with TensorFlow 2.0.
- **Logistic Regression** ([notebook](#)). Implement a Logistic Regression with TensorFlow 2.0.
- **Word2Vec (Word Embedding)** ([notebook](#)). Build a Word Embedding Model (Word2Vec) from Wikipedia data, with TensorFlow 2.0.

3 - Neural Networks

Supervised

- **Simple Neural Network** ([notebook](#)). Use TensorFlow 2.0 'layers' and 'model' API to build a simple neural network to classify MNIST digits dataset.
- **Simple Neural Network (low-level)** ([notebook](#)). Raw implementation of a simple neural network to classify MNIST digits dataset.
- **Convolutional Neural Network** ([notebook](#)). Use TensorFlow 2.0 'layers' and 'model' API to build a convolutional neural network to classify MNIST digits dataset.

- **Convolutional Neural Network (low-level)** ([notebook](#)). Raw implementation of a convolutional neural network to classify MNIST digits dataset.
- **Recurrent Neural Network (LSTM)** ([notebook](#)). Build a recurrent neural network (LSTM) to classify MNIST digits dataset, using TensorFlow 2.0 'layers' and 'model' API.
- **Bi-directional Recurrent Neural Network (LSTM)** ([notebook](#)). Build a bi-directional recurrent neural network (LSTM) to classify MNIST digits dataset, using TensorFlow 2.0 'layers' and 'model' API.
- **Dynamic Recurrent Neural Network (LSTM)** ([notebook](#)). Build a recurrent neural network (LSTM) that performs dynamic calculation to classify sequences of variable length, using TensorFlow 2.0 'layers' and 'model' API.

Unsupervised

- **Auto-Encoder** ([notebook](#)). Build an auto-encoder to encode an image to a lower dimension and re-construct it.
- **DCGAN (Deep Convolutional Generative Adversarial Networks)** ([notebook](#)). Build a Deep Convolutional Generative Adversarial Network (DCGAN) to generate images from noise.

4 - Utilities

- **Save and Restore a model** ([notebook](#)). Save and Restore a model with TensorFlow 2.0.
- **Build Custom Layers & Modules** ([notebook](#)). Learn how to build your own layers / modules and integrate them into TensorFlow 2.0 Models.

5 - Data Management

- **Load and Parse data** ([notebook](#)). Build efficient data pipeline with TensorFlow 2.0 (Numpy arrays, Images, CSV files, custom data, ...).
- **Build and Load TFRecords** ([notebook](#)). Convert data into TFRecords format, and load them with TensorFlow 2.0.
- **Image Transformation (i.e. Image Augmentation)** ([notebook](#)). Apply various image augmentation techniques with TensorFlow 2.0, to generate distorted images for training.

TensorFlow v1

The tutorial index for TF v1 is available here: [TensorFlow v1.15 Examples](#). Or see below for a list of the examples.

Dataset

Some examples require MNIST dataset for training and testing. Don't worry, this dataset will automatically be downloaded when running examples. MNIST is a database of handwritten digits, for a quick description of that dataset, you can check [this notebook](#).

Official Website: <http://yann.lecun.com/exdb/mnist/>.

Installation

To download all the examples, simply clone this repository:

```
git clone https://github.com/aymericdamien/TensorFlow-Examples
```

To run them, you also need the latest version of TensorFlow. To install it:

```
pip install tensorflow
```

or (with GPU support):

```
pip install tensorflow_gpu
```

For more details about TensorFlow installation, you can check [TensorFlow Installation Guide](#)

TensorFlow v1 Examples - Index

The tutorial index for TF v1 is available here: [TensorFlow v1.15 Examples](#).

0 - Prerequisite

- [Introduction to Machine Learning](#).
- [Introduction to MNIST Dataset](#).

1 - Introduction

- **Hello World** ([notebook](#)) ([code](#)). Very simple example to learn how to print "hello world" using TensorFlow.

- **Basic Operations** ([notebook](#)) ([code](#)). A simple example that cover TensorFlow basic operations.
- **TensorFlow Eager API basics** ([notebook](#)) ([code](#)). Get started with TensorFlow's Eager API.

2 - Basic Models

- **Linear Regression** ([notebook](#)) ([code](#)). Implement a Linear Regression with TensorFlow.
- **Linear Regression (eager api)** ([notebook](#)) ([code](#)). Implement a Linear Regression using TensorFlow's Eager API.
- **Logistic Regression** ([notebook](#)) ([code](#)). Implement a Logistic Regression with TensorFlow.
- **Logistic Regression (eager api)** ([notebook](#)) ([code](#)). Implement a Logistic Regression using TensorFlow's Eager API.
- **Nearest Neighbor** ([notebook](#)) ([code](#)). Implement Nearest Neighbor algorithm with TensorFlow.
- **K-Means** ([notebook](#)) ([code](#)). Build a K-Means classifier with TensorFlow.
- **Random Forest** ([notebook](#)) ([code](#)). Build a Random Forest classifier with TensorFlow.
- **Gradient Boosted Decision Tree (GBDT)** ([notebook](#)) ([code](#)). Build a Gradient Boosted Decision Tree (GBDT) with TensorFlow.
- **Word2Vec (Word Embedding)** ([notebook](#)) ([code](#)). Build a Word Embedding Model (Word2Vec) from Wikipedia data, with TensorFlow.

3 - Neural Networks

Supervised

- **Simple Neural Network** ([notebook](#)) ([code](#)). Build a simple neural network (a.k.a Multi-layer Perceptron) to classify MNIST digits dataset. Raw TensorFlow implementation.
- **Simple Neural Network (tf.layers/estimator api)** ([notebook](#)) ([code](#)). Use TensorFlow 'layers' and 'estimator' API to build a simple neural network (a.k.a Multi-layer Perceptron) to classify MNIST digits dataset.
- **Simple Neural Network (eager api)** ([notebook](#)) ([code](#)). Use TensorFlow Eager API to build a simple neural network (a.k.a Multi-layer Perceptron) to classify MNIST digits dataset.

- **Convolutional Neural Network** ([notebook](#)) ([code](#)). Build a convolutional neural network to classify MNIST digits dataset. Raw TensorFlow implementation.
- **Convolutional Neural Network (tf.layers/estimator api)** ([notebook](#)) ([code](#)). Use TensorFlow 'layers' and 'estimator' API to build a convolutional neural network to classify MNIST digits dataset.
- **Recurrent Neural Network (LSTM)** ([notebook](#)) ([code](#)). Build a recurrent neural network (LSTM) to classify MNIST digits dataset.
- **Bi-directional Recurrent Neural Network (LSTM)** ([notebook](#)) ([code](#)). Build a bi-directional recurrent neural network (LSTM) to classify MNIST digits dataset.
- **Dynamic Recurrent Neural Network (LSTM)** ([notebook](#)) ([code](#)). Build a recurrent neural network (LSTM) that performs dynamic calculation to classify sequences of different length.

Unsupervised

- **Auto-Encoder** ([notebook](#)) ([code](#)). Build an auto-encoder to encode an image to a lower dimension and re-construct it.
- **Variational Auto-Encoder** ([notebook](#)) ([code](#)). Build a variational auto-encoder (VAE), to encode and generate images from noise.
- **GAN (Generative Adversarial Networks)** ([notebook](#)) ([code](#)). Build a Generative Adversarial Network (GAN) to generate images from noise.
- **DCGAN (Deep Convolutional Generative Adversarial Networks)** ([notebook](#)) ([code](#)). Build a Deep Convolutional Generative Adversarial Network (DCGAN) to generate images from noise.

4 - Utilities

- **Save and Restore a model** ([notebook](#)) ([code](#)). Save and Restore a model with TensorFlow.
- **Tensorboard - Graph and loss visualization** ([notebook](#)) ([code](#)). Use Tensorboard to visualize the computation Graph and plot the loss.
- **Tensorboard - Advanced visualization** ([notebook](#)) ([code](#)). Going deeper into Tensorboard; visualize the variables, gradients, and more...

5 - Data Management

- **Build an image dataset** ([notebook](#)) ([code](#)). Build your own images dataset with TensorFlow data queues, from image folders or a dataset file.

- **TensorFlow Dataset API** ([notebook](#)) ([code](#)). Introducing TensorFlow Dataset API for optimizing the input data pipeline.
- **Load and Parse data** ([notebook](#)). Build efficient data pipeline (Numpy arrays, Images, CSV files, custom data, ...).
- **Build and Load TFRecords** ([notebook](#)). Convert data into TFRecords format, and load them.
- **Image Transformation (i.e. Image Augmentation)** ([notebook](#)). Apply various image augmentation techniques, to generate distorted images for training.

6 - Multi GPU

- **Basic Operations on multi-GPU** ([notebook](#)) ([code](#)). A simple example to introduce multi-GPU in TensorFlow.
- **Train a Neural Network on multi-GPU** ([notebook](#)) ([code](#)). A clear and simple TensorFlow implementation to train a convolutional neural network on multiple GPUs.