DIVE INTO DEEP LEARNING

TREVOR YU & CARTER DEMARS

TODAY'S AGENDA

- Review introductory deep learning concepts from the Neural Networks for Novices workshop.
- Develop intuition for high-level neural network architectures for tasks in computer vision, natural language, and signal processing.
- Investigate how deep learning can be applied to different data modalities and for a wider variety of machine learning tasks in industry and academia.
- Revisit the MNIST example from last workshop, in PyTorch.
- Apply this knowledge to implement a deep learning model in PyTorch, based on technical specifications from a research paper.

CLASSICAL ML: KEY TAKEAWAYS

- Supervised learning requires pairs of input data and ground-truth labels to train a machine learning model to learn patterns from said data
- We train our models on the training dataset to update parameters, then test the performance on a testing dataset to see how well it can generalize
- We only care about performance on the test dataset of unseen examples
- More complex models have high variance and can overfit to random noise in the training data, reducing generalization on test data

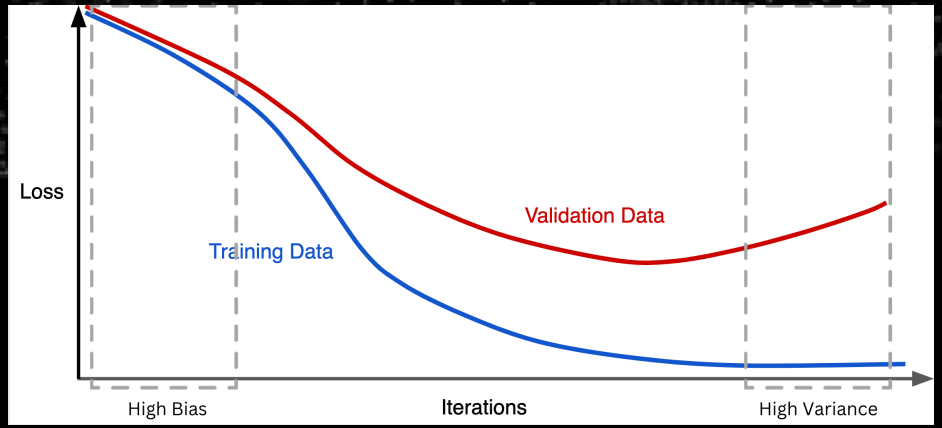
NEURAL NETWORKS FOR NOVICES: KEY TAKEAWAYS

- Neural networks form a highly versatile category of machine learning models that can learn complex, non-linear relationships from data
- They are widely used for tasks in computer vision, anomaly detection, natural language processing, signal processing, and time series forecasting
- Recall that neural networks operate on tensors, including n-dimensional arrays such as vectors and matrices

- Neural networks apply mathematical transformations to the input based on weights that are "learned" using optimization algorithms such as gradient descent
- Through the process of repeated forward and backpropagation, the network updates its parameters. This process is regulated by hyperparameters, such as learning rate, selected prior to training.

REGULARIZATION

- Recall that when a model overfits, it fails to generalize to unseen data
- A high variance model ends up capturing the noise in the training set, which is detrimental to its performance on the validation set
- Regularization techniques can discourage your model from overfitting to the training set, by penalizing large weights and minimizing the model's dependence on any one feature



L2 REGULARIZATION

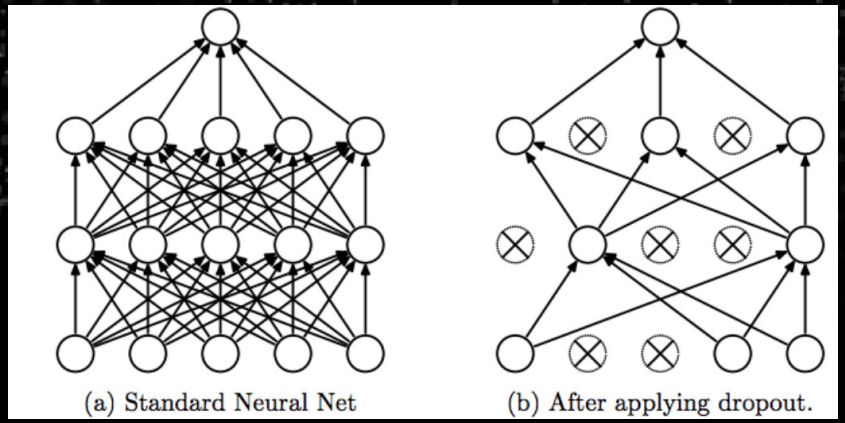
- Regularization penalizes high weights within the neural network during training, and so each individual neuron has a smaller effect on the result, reducing overfitting.
- Add a scaled regularization term using the Frobenius Norm of a matrix (sum of squared elements)
- Lambda is the regularization parameter (a hyperparameter) that is set using the validation set

L2 Regularization

$$\text{Modified loss function} = \text{Loss function} + \lambda \sum_{i=1}^{n} W_i^2$$

DROPOUT REGULARIZATION

- Randomly eliminating a percentage of nodes during each round of training
- It reduces the model's dependency on any single feature, so weights are more evenly distributed during training
- Formally, dropout regularization and L2 regularization both shrink the squared norm of the weight vectors

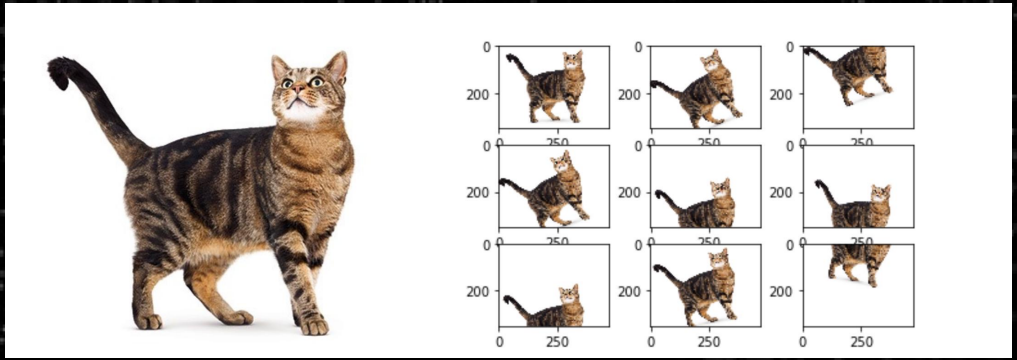


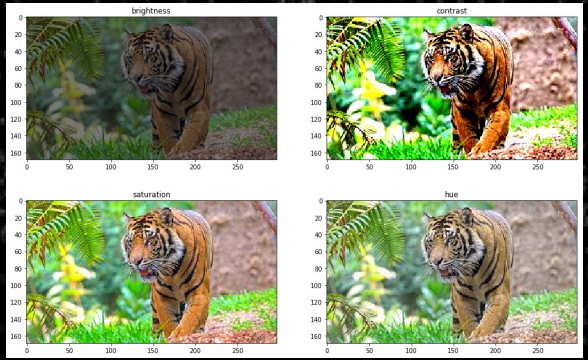
EARLY STOPPING

- Stopping the training of a neural network before overfitting occurs also prevents the weights from being updated to large values, so this is a form of regularization
- One downside of early stopping is that it we might stop training too soon, since it is difficult to simultaneously optimize the cost function while avoiding overfitting at the same time
  - Using L2 and dropout regularization makes the solution space for the hyperparameters much easier to search over, since you can train the network as long as possible
  - You would, however, need to test multiple values of lambda, which makes the process more computationally expensive

DATA AUGMENTATION

- For example, in computer vision, you can take transformations of existing training examples such as cropping or inverting images to “augment” your training dataset





- Increasing the size of the training set can reduce the likelihood of overfitting

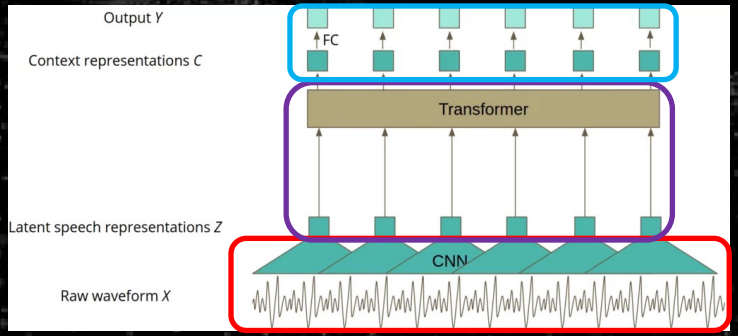
INTUITION: LAYER TYPES

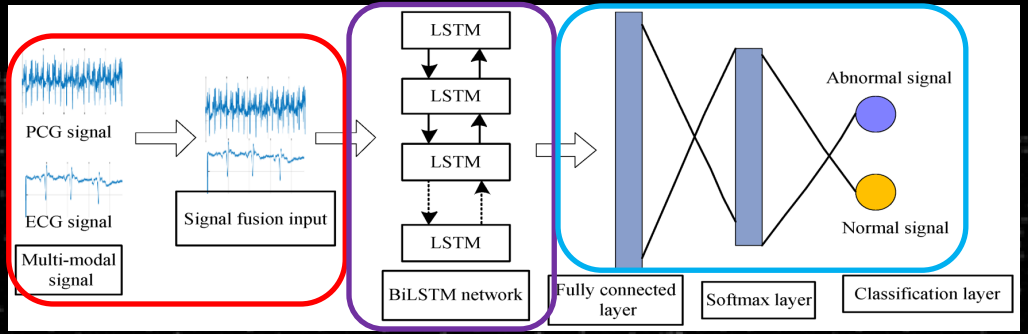
- Linear
  - Linear mixing of features from previous layer
  - Combined with non-linear layer, creates an adaptable, non-linear feature extractor
  - Majority of parameters
  - E.g. Linear/dense/fully connected, convolution, attention (sort of)
- Non-linear
  - Introduces flexibility to model non-linear relationships
  - Not typically parameterized
  - E.g. ReLU, tanh, sigmoid, GELU, swish
- Regularization
  - Imposes restrictions on model to reduce complexity without playing with many architecture hyperparameters
  - Few parameters
  - E.g. BatchNorm, LayerNorm, Dropout

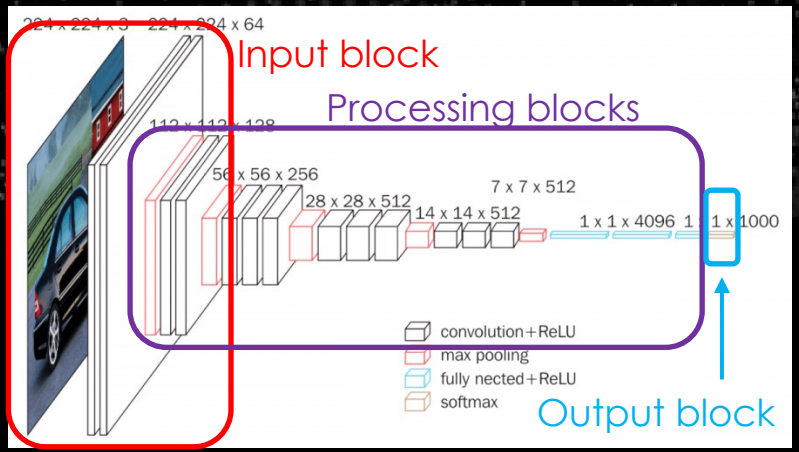
INTUITION: HIGH-LEVEL ARCHITECTURES

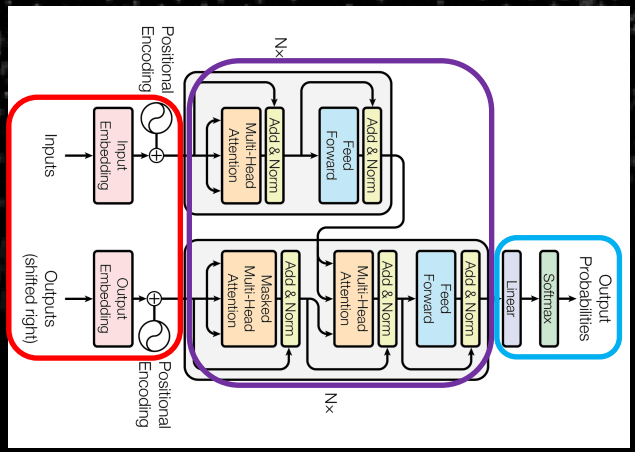
- Blocks of common layer structures are stacked to create deep network
  - Typically blocks contain some linear, 1 non-linear, and some regularization layers
- Input block transforms inputs to compatible shape
- Processing blocks are non-linear feature extractors in a hidden (latent) space
- Output blocks
  - Last layer is regression / classification on these complex features
  - Can have multiple output "heads" to do different tasks
- Architectures often taken from research papers, but popular architectures are available in Keras, torchvision, Huggingface Transformers, etc.

INPUT-PROCESSING-OUTPUT



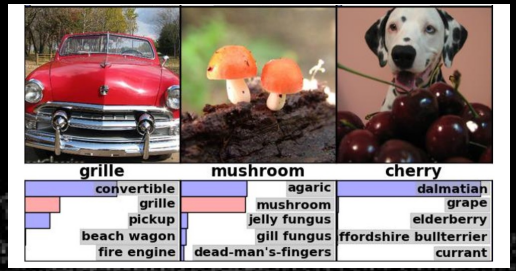


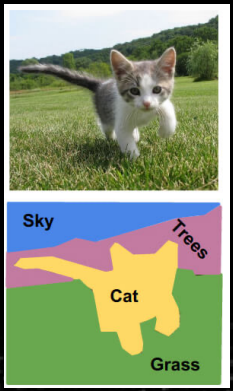


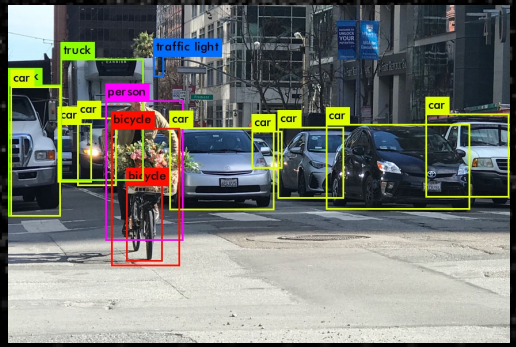


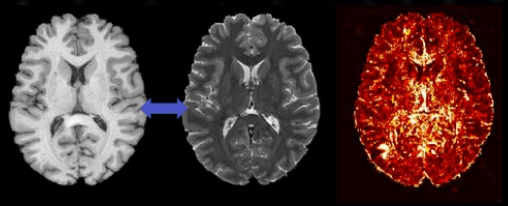
COMPUTER VISION

- Common tasks:
  - Image classification
  - Object detection (presence, location, size)
  - Motion tracking
  - Pose estimation
  - Scene reconstruction
- Many kinds of images: Anything with spatial data (2-D, sometimes 3-D)
  - Natural images (e.g. cameras, RGB)
  - Medical images (MRI, CT scan)
  - Different wavelengths (Radio, Microwave, IR, UV, X-ray)
  - Remote sensing / satellite



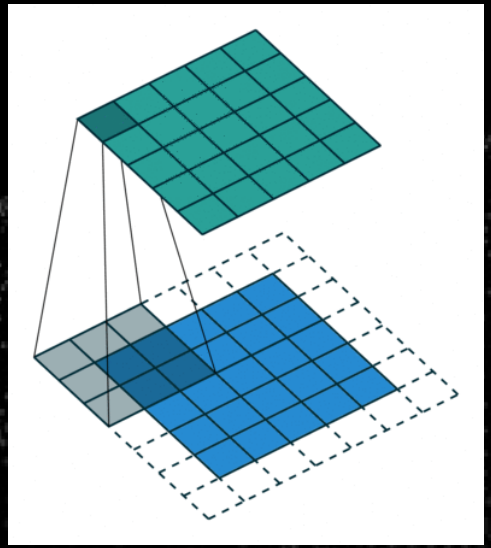






PROCESSING IMAGES WITH NEURAL NETWORKS

- Images are easily representable as size (W, H, C) tensors
  - Each pixel is a C-dimensional vector
  - Often have to crop to a fixed size
- Networks use convolutional layers
  - Assumes images have meaningful correlations with local pixels
  - Fewer parameters than a linear layer
  - Convolution operations is like sliding a small weight matrix (kernel) along spatial dims and combining information from across channels
  - Most modern networks use padding to achieve the "same" output size, and use strided convolutions instead of pooling



MULTI-CHANNEL CONVOLUTION

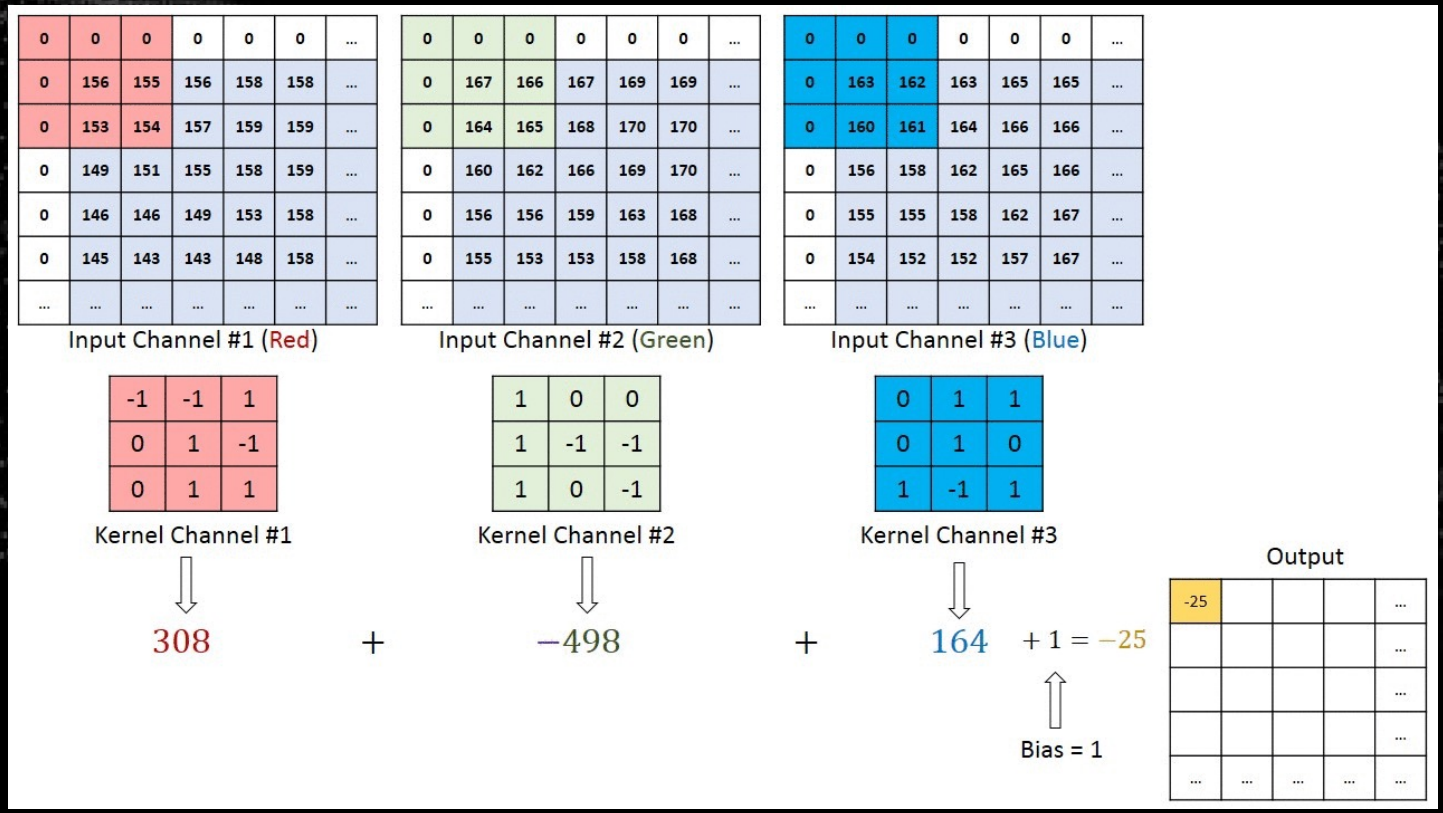
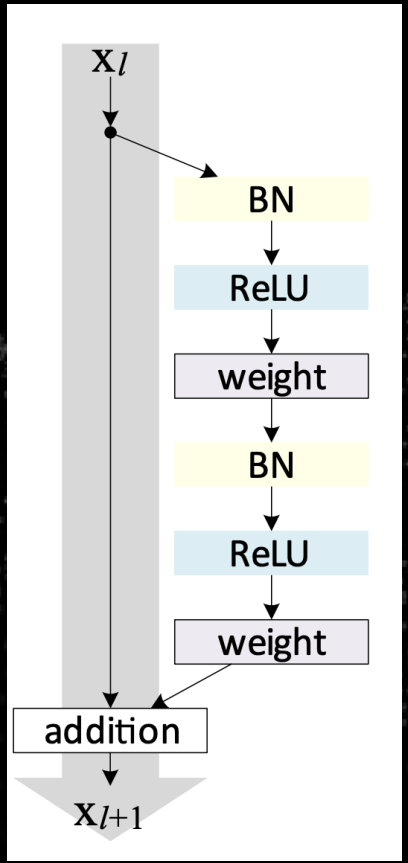


IMAGE PROCESSING ARCHITECTURES

- Backbone-Neck-Head
  - Backbone is feature extractor from inputs
  - Neck condenses features for head (typically pooling or some sort of combination of lower-level features)
  - Head performs task (e.g. regression or classification)
- Feature extraction makes a compressed representation deeper into the network
  - Reduction in spatial size
  - Increase in feature dimension/channels
- Notable backbone architectures: ResNet, MobileNet, U-Net, ViT

EXAMPLE BLOCK

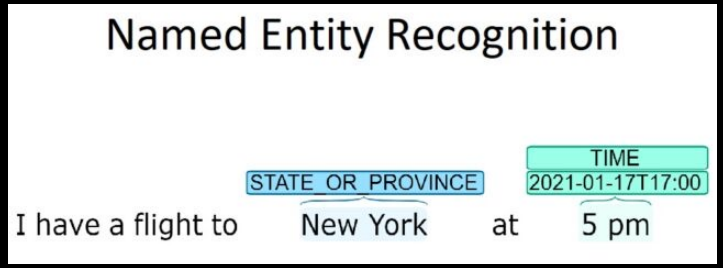
- ResNetV2 Block
  - Linear: Convolution (weight)
  - Non-linear: ReLU
  - Regularization: BatchNorm (BN)
- We will implement this at the end of the workshop

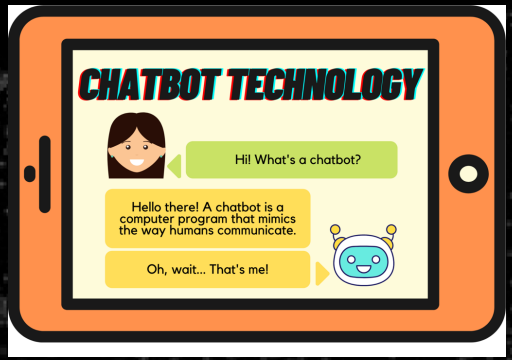


ResNetV2 block
(He et al., 2016)

NATURAL LANGUAGE PROCESSING

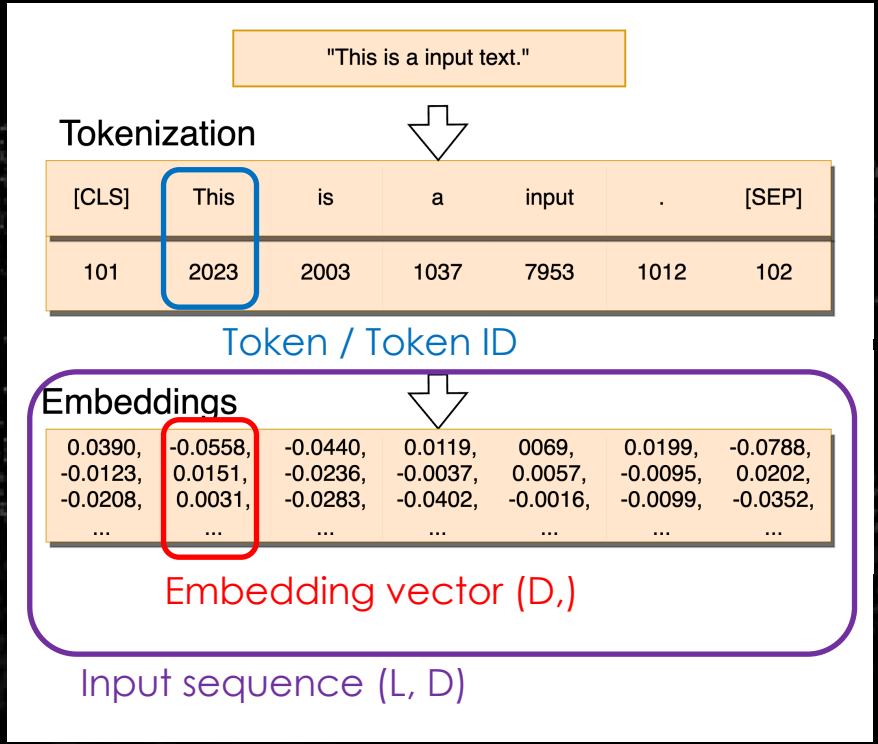
- Common tasks
  - Text classification
  - Word-level classification (named entity recognition)
  - Text extraction (Question-answering)
  - Text generation
    - Chat and command
    - Summarization
- Modalities
  - Most models trained on English text, but others have multi-lingual capacities
  - Books, social media, website content, conversations
  - Code, serialized data (e.g. JSON)
  - From many specialized domains (e.g. medical, financial, legal, etc.)





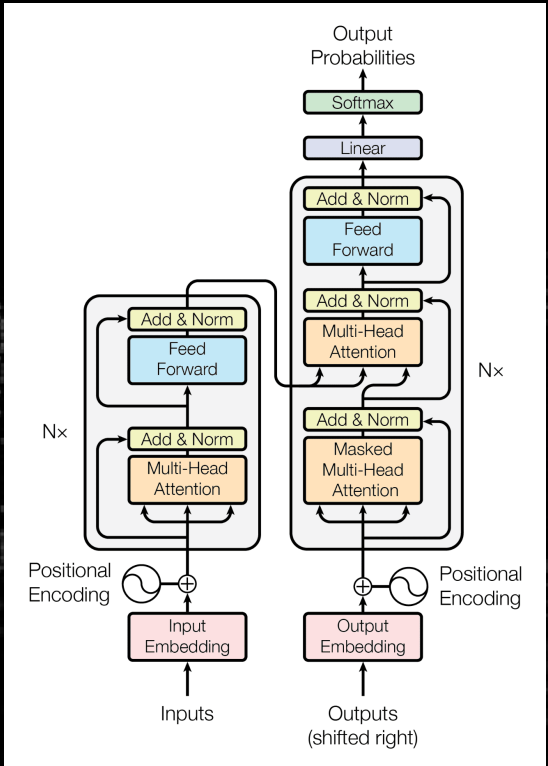
PROCESSING TEXT WITH NEURAL NETWORKS

- How to convert text into vectors?
  - Convert text into sub-word pieces called “tokens”
  - Finite vocabulary of tokens each has an associated “embedding vector”
  - Text is converted into a sequence of vectors of shape (L, D)
  - NNs like constant size inputs – pad with 0s up to a maximum length
- Some early success with LSTMs but now transformer networks are dominant in NLP



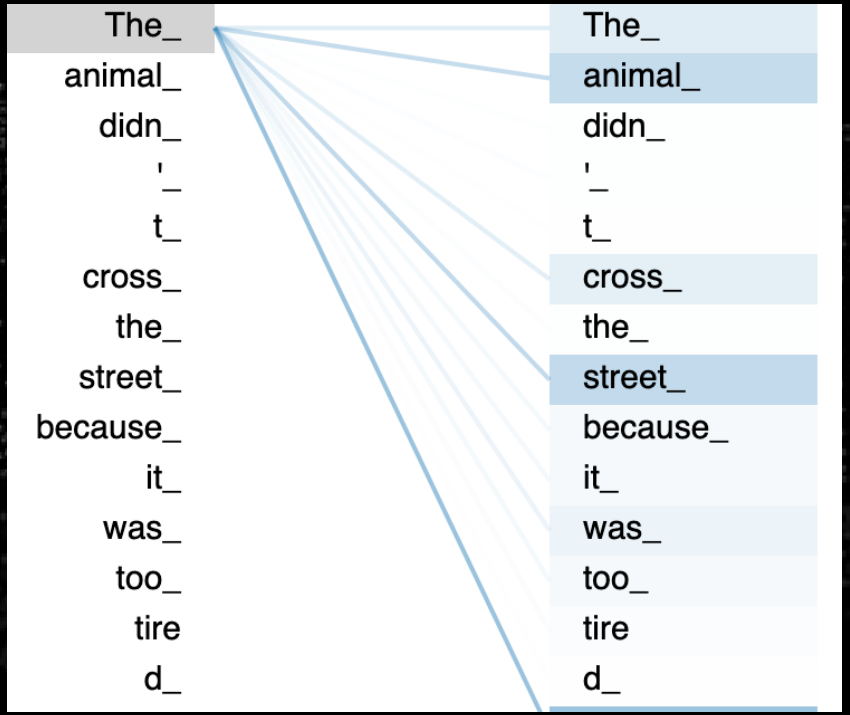
NLP ARCHITECTURES

- Embedding input layer
  - Lookup token sequence
  - Add information about positions / sequence order
- Transformer processing layer
  - Transformers alternate between attention blocks and feed-forward network blocks
- Output layers
  - Classification over vocabulary of tokens to predict next word (GPT)
  - Classification over entire sentence or for each word (BERT)

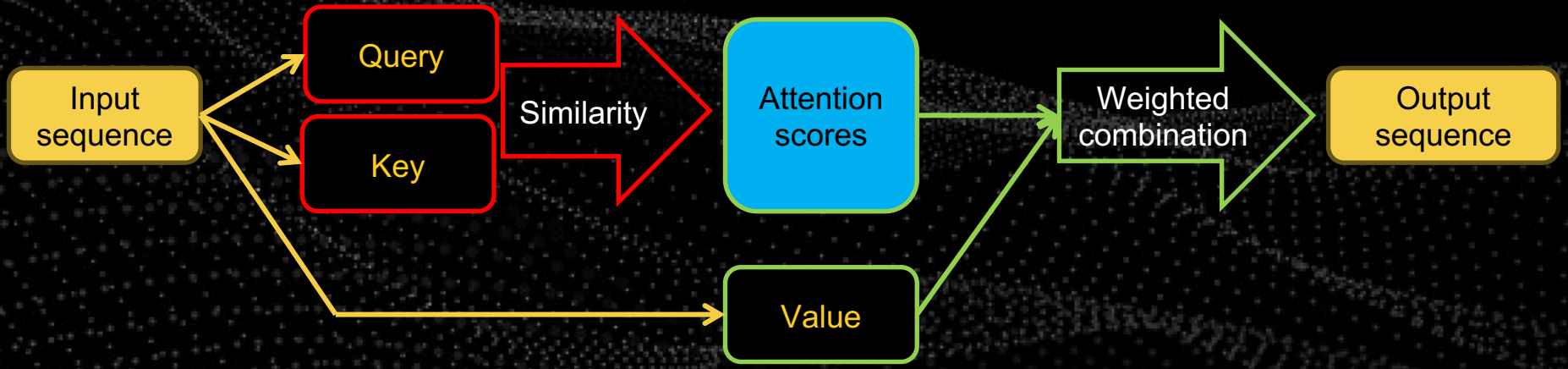


ATTENTION

- Attention allows for contextual representations to be built
- Compute similarity of sequence elements to each other
- Information from highly similar elements are combined in a weighted sum



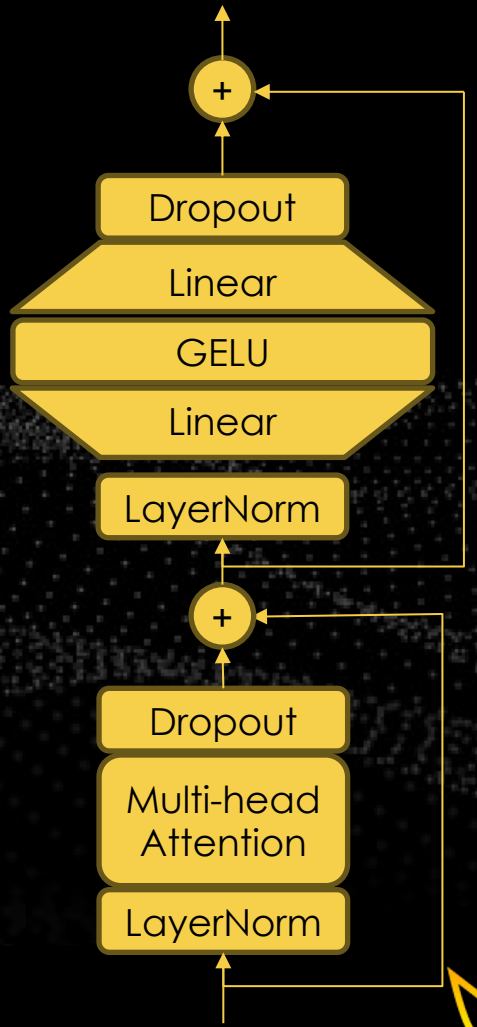
ATTENTION



$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

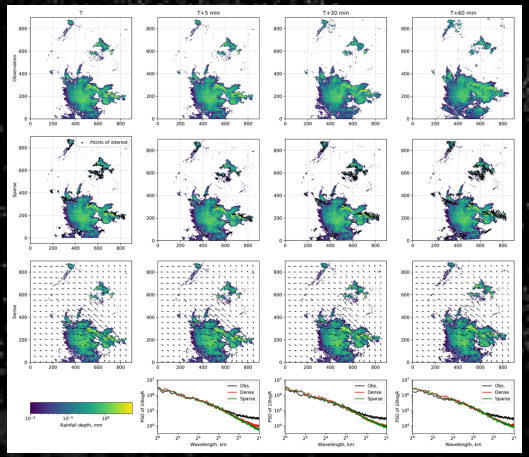
EXAMPLE TRANSFORMER BLOCK

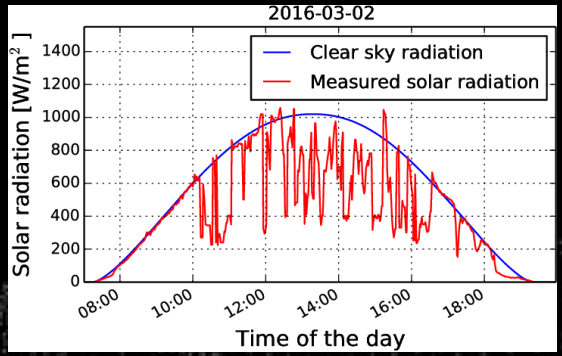
- Linear: Attention*, Linear
- Non-linear: GELU, softmax (in attention)
- Regularization: LayerNorm, Dropout

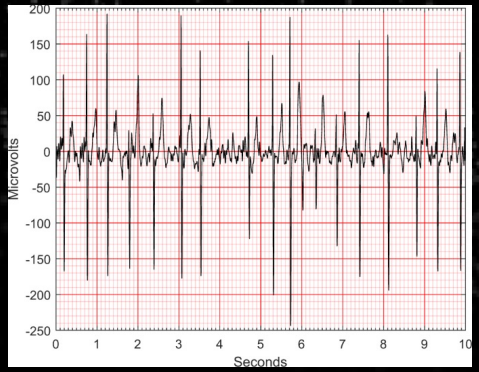


SIGNAL PROCESSING & TIME SERIES FORECASTING

- Common tasks
  - Time series forecasting
  - Weather prediction
  - Stock price prediction
  - Speech recognition
  - Signal classification
  - Anomaly detection
- Common modalities
  - Bio-signals
  - Sensor data
  - Video and audio







PROCESSING SIGNALS WITH NEURAL NETWORKS

- Signals (time-series) are already numerical, but may need to be resampled to regular intervals
- Recurrent networks and transformers often used, sometimes CNNs too

Shazam uses signal processing techniques in combination with a certain type of RNN to match songs to user audio.

SIGNAL PROCESSING ARCHITECTURES: LSTM

- Long Short-Term Memory Networks (LSTMs) are a type of Recurrent Neural Network (RNN) that is suitable for modelling long-range dependencies and sequential data, such as time series or or natural language
- LSTM networks have memory cells that can store and update information over time.
- They use three gates (input gate, forget gate, and output gate) to control the flow of information through the cell

PyTorch vs Tensorflow

- Compared to Tensorflow, PyTorch:
  - Has more control and flexibility for the developer
  - Requires more code to be written
  - Data needs to be manually sent to GPU devices
  - More of an object-oriented design pattern
  - Tends to be used more in research than industry
- Libraries like PyTorch Lightning and Huggingface transformers introduce trainer APIs that are compatible with PyTorch models
- Tensorflow's lower-level API underneath Keras is very similar to PyTorch

TRAINING A NETWORK

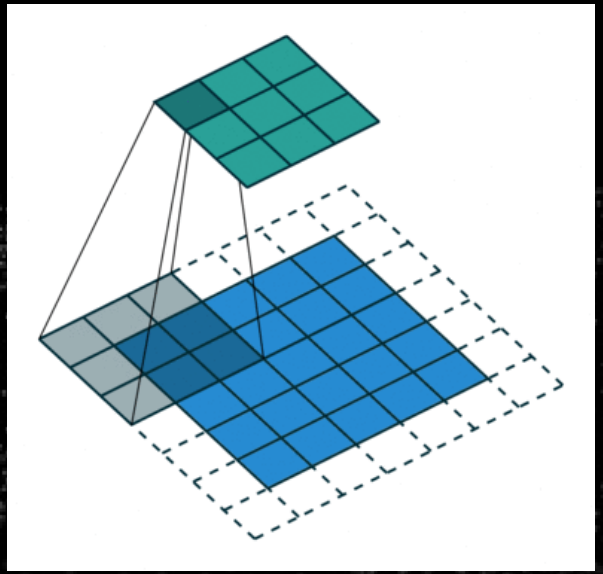
1. Obtain dataset
2. Convert data to tensors
3. Define model architecture
4. Select hyperparameters
5. Fit model with gradient descent
    1. Fetch next batch of data
    2. Reset gradients
    3. Forward pass with input data
    4. Compute loss on model predictions and labels
    5. Compute gradients with backpropagation
    6. Update parameters with gradient descent step
6. Evaluate on test dataset

CODE-ALONG: MNIST MLP

- Model architecture is set in `__init__`
  - Name and assign other `nn.Module` layers to class attributes
- Model computation is defined in `forward`
  - Defines flow of data from inputs through layers defined in `__init__` to compute model output

CODE-ALONG: RESNET-V2 MODEL

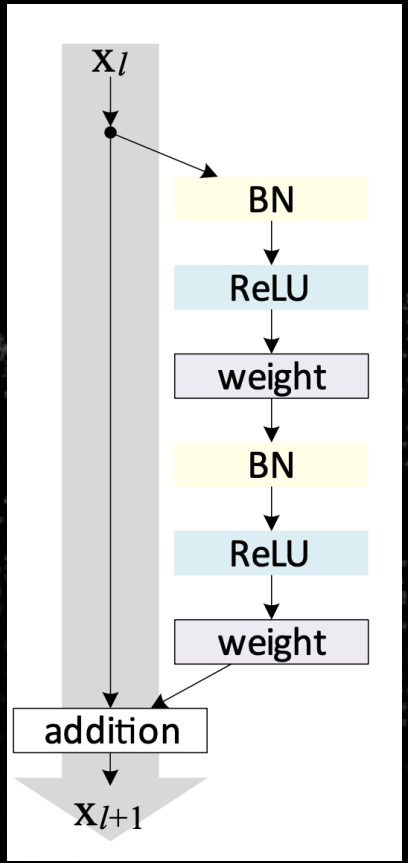
- Recall the structure of this ResNet-V2 block from earlier
- We will code a model based on the ResNet-V2 block in PyTorch
- Convolution operation arguments
  - Number of input and output channels
  - Stride (downsampling of spatial dimensions)
  - Kernel size (3x3)
  - Padding (1)
- Convolution effect on shapes
  - Input: (B, C_{in}, H, W)
  - Output: (B, C_{out}, H/stride, W/stride)



Stride conv reduces spatial dims

CODE-ALONG: RESNET-V2 MODEL

- Input layer:
  - Conv(1, 32)
  - BatchNorm
- Processing layers:
  - ResNetBlock(32, 32)
  - ResNetBlock(32, 64, s=2)
  - ResNetBlock(64, 64)
  - ResNetBlock(64, 128, s=2)
  - ResNetBlock(128, 128)
- Output layers:
  - AdaptiveAveragePooling
  - Linear(128, 10)



ResNetV2 block
(He et al., 2016)

UPCOMING EDUCATION SESSIONS

- Special topics sessions TBD
- Discord:
- https://discord.gg/46KUMNGE8J
- Will post recordings, slides, etc.