CONVOLUTIONAL NEURAL NETWORKS

COREY ADAMS
Computational Scientist
Physicist

12/10/23
San Juan, PR

ABOUT ME

- Computational Scientist at Argonne National Laboratory
- Joint Appointment between the Argonne Leadership Computing Facility and the Physics Division.
- Research interests are AI + fundamental physics, high performance deep learning (for science!) and neutrinoless double beta decay.
- Please interrupt with questions along the way!

DISCLAIMER

- Examples in these talks skew towards my own work and applications.
  - There are **so many** high quality applications of CNNs and GNNs in particle physics.
  - I've done this so I can include source code links and answer technical questions so please ask!
- I won't talk much about some other areas of my work but feel free to ask afterwards any time:
  - Sparse IO
  - CNN computational performance
  - Distributed training of CNNs and GNNs
  - (Other high performance computing things)

CONVOLUTIONAL NEURAL NETWORKS: A HISTORY IN 5 MINUTES

IMAGENET AND ALEXNET

In the late 2000's (before 2010), a researcher produced a dataset based on real-world images, with labels, called **ImageNet**. Synonymous with a yearly competition for the best algorithm. Originated by Fei-Fei Li.

AlexNet is the winning entry of the Large Scale Visual Recognition Challenge and a publication with now more than 120k citations, 10 years later.

Article on ImageNet

IMAGENET AND ALEXNET

- AlexNet back then:

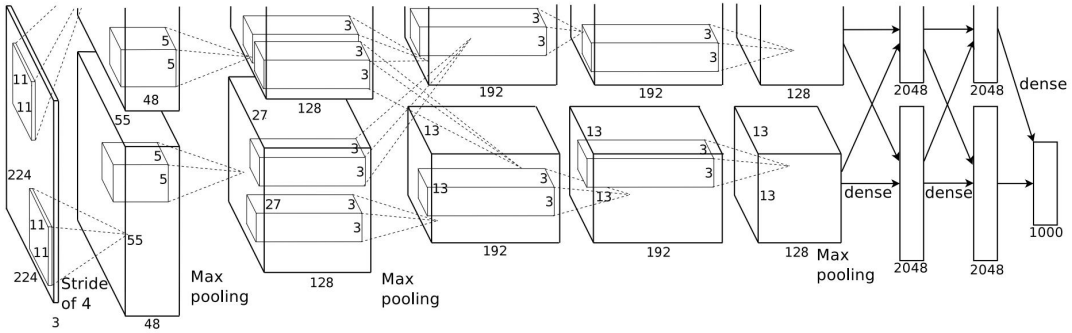


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

AlexNet

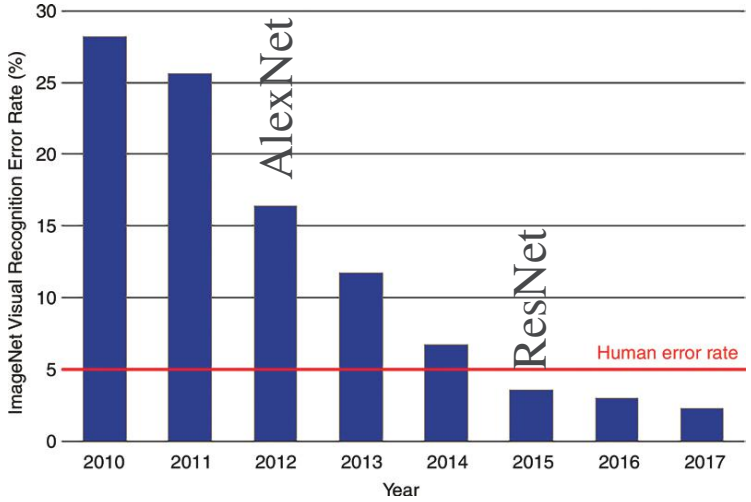
ImageNet: A large-scale hierarchical image database

IMAGENET AND ALEXNET

- AlexNet back then:
  - No Torch/TF/JAX
  - Written directly in CUDA.
  - Not the first for convolutions, pools, ReLU, etc., but one of the first to put the pieces together
  - Took “five to six days” to train.
- Today:

```
import torch
model = torch.hub.load('pytorch/vision:v0.10.0', 'alexnet', pretrained=True)
model.eval()
```

IMAGENET AND ALEXNET



Reference

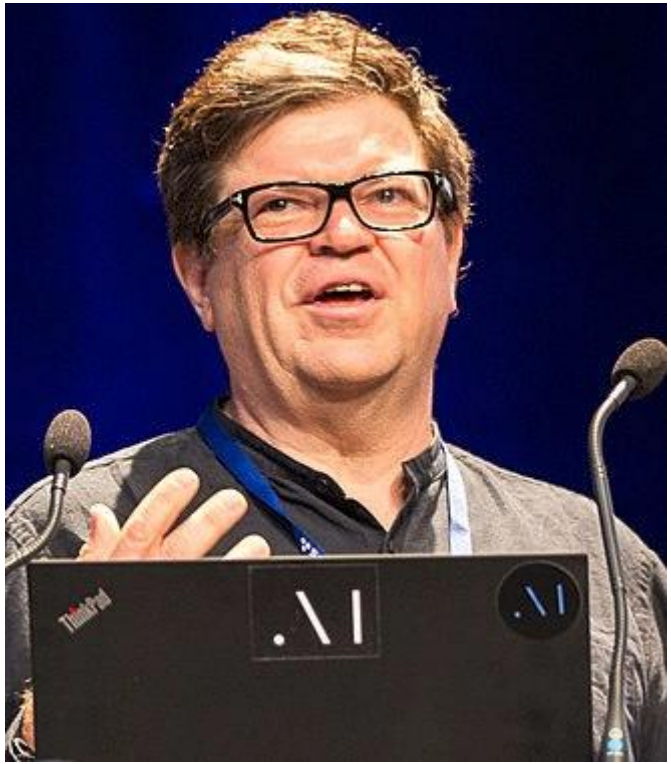
ImageNet drove advancement of computer vision quickly past human error rate.

The competitive nature + massive curated dataset + growing ease-of-GPU drove major theoretical and practical advances in image recognition, with **convolutional neural networks** at the heart.

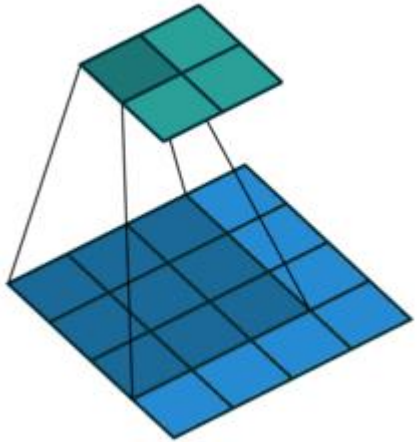
(Many ImageNet winners are now big-shots at AI companies and still in the news today)

WHAT ARE CONVOLUTIONS?

- First big demonstration by Yann LeCun in LeNet.
  - 1989 prototype!
  - One of the first applications of the backprop alogirhtm.
- Today, convolutions are one of the most dominant components of image models.
  - Yes, transformers work too, see Kazu's talk after lunch.



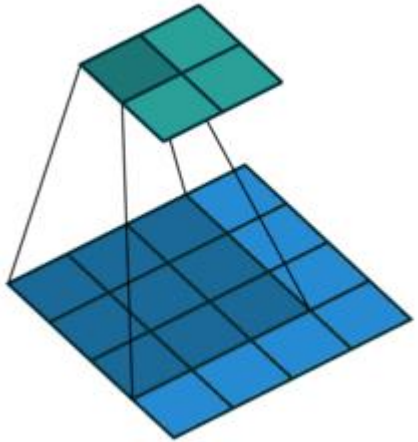
WHAT ARE CONVOLUTIONS?



Source

- Convolutions are a simple technique to transform an image, focused on **localality** as a driving factor.
- In principle, a convolution could be reproduced by just "flattening" the image and throwing it into a dense neural network layer.
- Forcing weight reuse over a small area over the entire image is a **bias** in training: the network is forced to focus on aggregating local, and only local, features.
- Multiple convolutional layers create a "receptive field"

CONVOLUTION MATH



Source

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$$

- Note that CNNs have multiple “channels” analagous to RGB channels. Convolutions look at all filters at once (by default) and have a separate learned filter for each output channel.
- Some limiting cases:
  - A 1x1 convolution over an input with k filters is mathematically the same as the same MLP over each pixel.
  - A [m,n] filter over an [m,n] image, with no padding and stride 1, is a fully connected neural network.

https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9

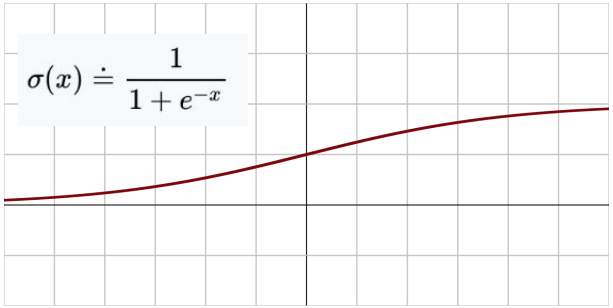
CNN: CORE INGREDIENTS

- Most convolutional networks employ additional techniques: normalizations, activations, and pooling layers.
  - **Activation**: The nonlinearity required to make the network into a universal approximator.
  - **Normalization**: mapping of the data (activations, conv outputs, etc) into a standard mean and variance.
    - Shown to dramatically reduce overfitting and improve convergence of networks.
  - **Pooling**: Any operation that reduces the spatial size of the inputs.

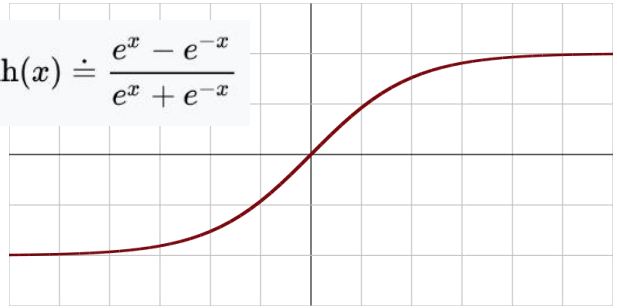
ACTIVATIONS

- Non-linear activation functions are essential for turning neural networks into universal function approximators. Generally two classes:
  - **Saturating:** the activation function reaches a maximum value as x goes to infinity
    - Examples: sigmoid, tanh.
    - Problem: As the input to the activation gets too large (or small) the gradient and "upstream" updates go to 0.

$$\sigma(x) \doteq \frac{1}{1+e^{-x}}$$

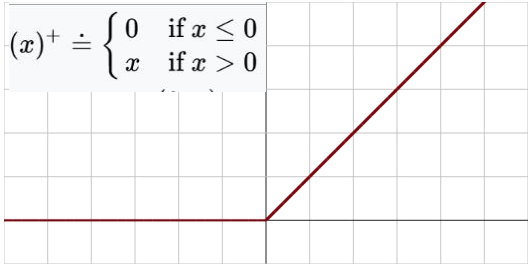


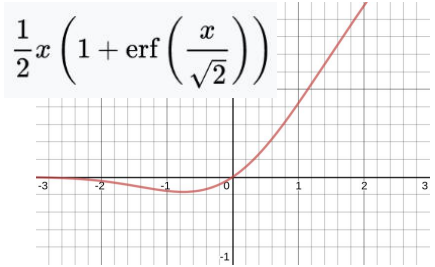
$$\tanh(x) \doteq \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$



ACTIVATIONS

- Non-linear activation functions are essential for turning neural networks into universal function approximators. Generally two classes:
  - **Non-saturating:** the activation function is unbounded in at least one direction (+/-).
    - Examples: ReLU, GeLU, Softplus, ...
    - These are the “standard” activations these days!
    - GeLU in particular is the “latest and greatest”

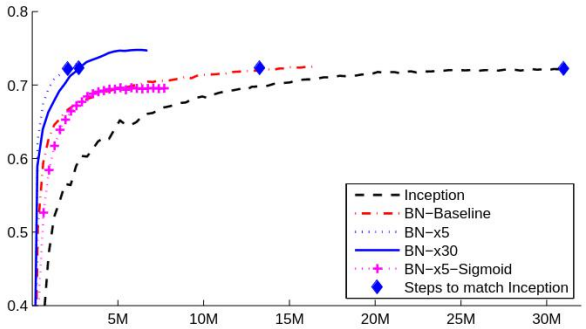




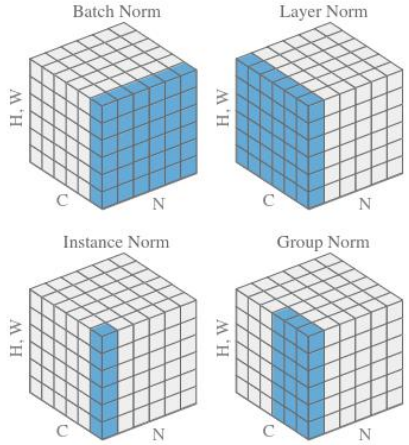
All images (and previous slide) from wikipedia.

NORMALIZATION

- Early neural networks struggled with “internal covariant shift” - all layers updated at once, but the inputs to any one layer were typically changing scale during training time.
- Batch Normalization, standardizing the mean and deviation of activations during training, allowed dramatic speed-up of training.



https://arxiv.org/pdf/1502.03167.pdf



https://arxiv.org/pdf/1903.10520.pdf

POOLING

- Convolutional layers leverage **pooling** layers as another critical **inductive bias**.
- To turn local features into global information, pooling aggregates local features while reducing spatial dimension sizes.
- **Max Pooling** keeps only the highest values
- **Average Pooling** keeps all ... averaged.
- (There are also more complicated poolings.)

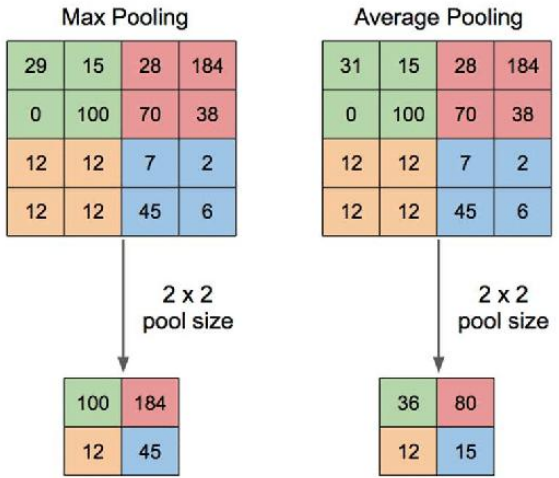
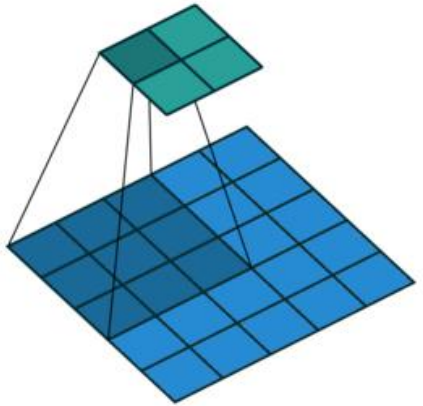


Image Source

POOLING

- Convolutional layers leverage **pooling** layers as another critical **inductive bias**.
- To turn local features into global information, pooling aggregates local features while reducing spatial dimension sizes.
- **Convolutions** with strides are also pooling!



Think of convolutional pooling as a learnable generalization of average pooling.

Generally, convolutions are “state of the art” downsampling.

VANISHING GRADIENTS

- Networks are trained with the **backpropogation** algorithm. AKA the chain rule.
- As networks get deeper, and particularly with **saturating activations**, the iterative update in the “earliest” parts of the network goes to 0 - “vanishing gradients.”
  - (This was actually one of the earliest challenges in training neural networks!)
- Many methods overcome this issue;
  - Normalization, and non-saturating activations, loss scaling
  - Gradient clipping solves the inverse problem (gradient explosion)
- One method stands out as a solution to the vanishing gradient problem: **Residual Networks**

RESIDUAL NETWORKS

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

- More citations than AlexNet!
- This is the paper that won ImageNet the year when machine learning beat human performance.
- https://arxiv.org/pdf/1512.03385.pdf

RESIDUAL NETWORKS

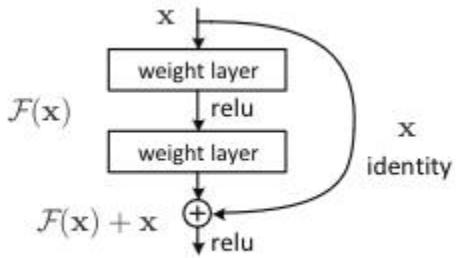
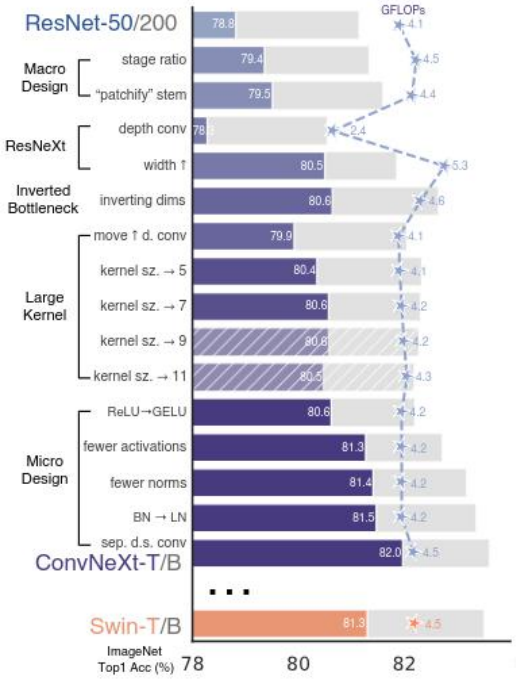


Figure 2. Residual learning: a building block.

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

- Residual networks **explicitly** encode layers to fit the "residual" of the input instead of the transformation of the input.
- At worst, if the gradient of the convolution goes to 0, the gradient of the whole residual layer goes to 1.
- Nearly all the best convolutional neural networks are using some form of skip / residual connection.

CONVNEXT: THE LATEST AND GREATEST?



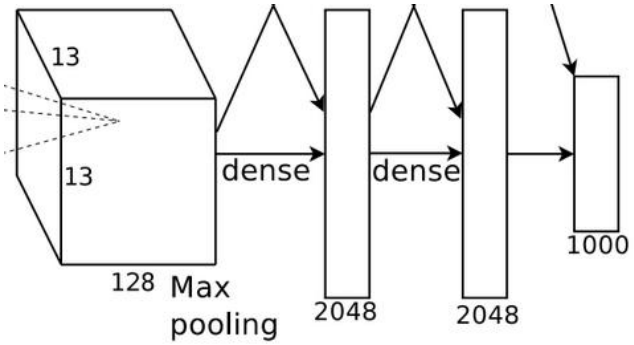
- A paper last year synthesized and systematically updated ResNet to match the accuracy of transformers on visual tasks.
- If you're active on image tasks and convolutional networks, it's worth a read and testing some of the ideas.
- Nearly all of the ideas in the paper are from previous publications that showed one technique is better/worse than another in vision - the synthesis leads to dramatic improvements.

https://arxiv.org/pdf/2201.03545.pdf

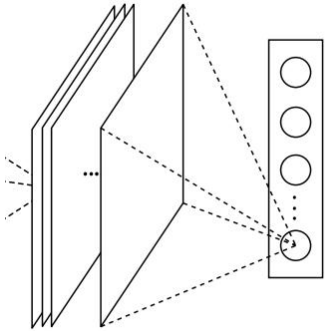
WHAT CAN YOU USE A CNN FOR?

CNN SPECIALIZATION: CLASSIFICATION

- Applying a Convolutional network for classification was the core task of imagenet.
- Somehow, after all the convolutional layers, you must turn the resulting image into a differentiable prediction.

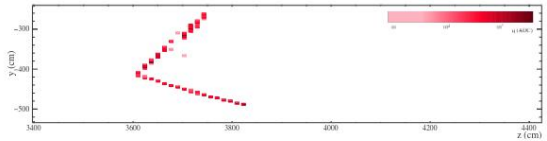


AlexNet used pooling, flattening, and dense layers to shape it's final classification

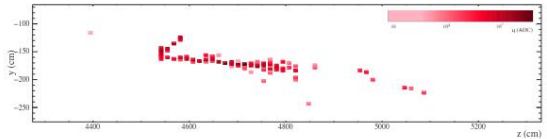


Network-in-Network used "global average pooling" after 1x1 bottleneck convolutions to remove the F.C. layers and shape the output correctly.

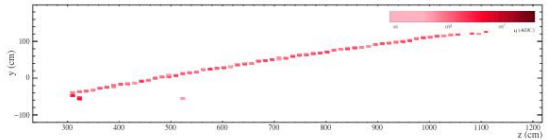
CLASSIFICATION EXAMPLE: NOVA



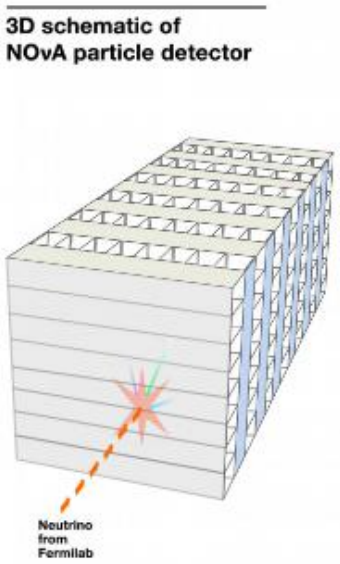
(a) A ν_e CC QE electron plus one hadron signature where the upward-going shower-like prong with multiple hit cells on each plane corresponds to an electron and the downward-going track-like prong with approximately one hit per plane correspond to a proton.

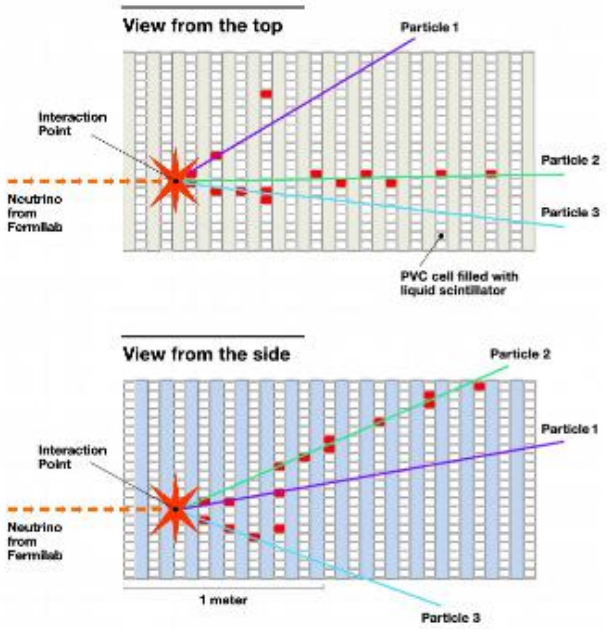


(b) A ν_e CC RES electron plus hadron shower signature with a characteristic electron shower and short prongs which could correspond to multiple hadrons.



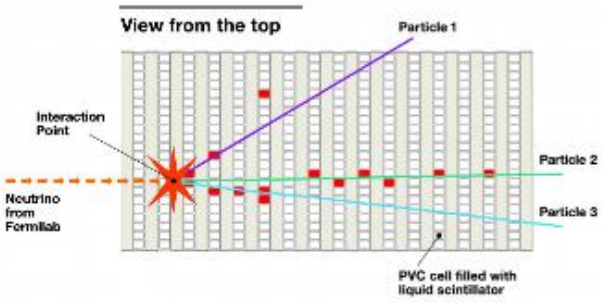
(c) A ν_μ CC QE muon plus one hadron signature with a long track-like prong with lower charger-per-cell corresponding to a muon and a short prong with larger charge-per-cell corresponding to a proton.

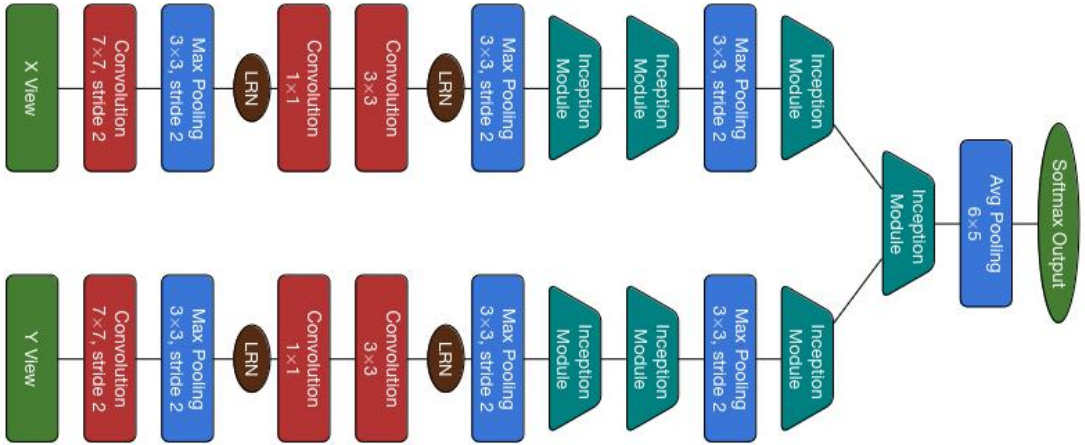




A Convolutional Neural Network Neutrino Event Classifier

CLASSIFICATION EXAMPLE: NOVA

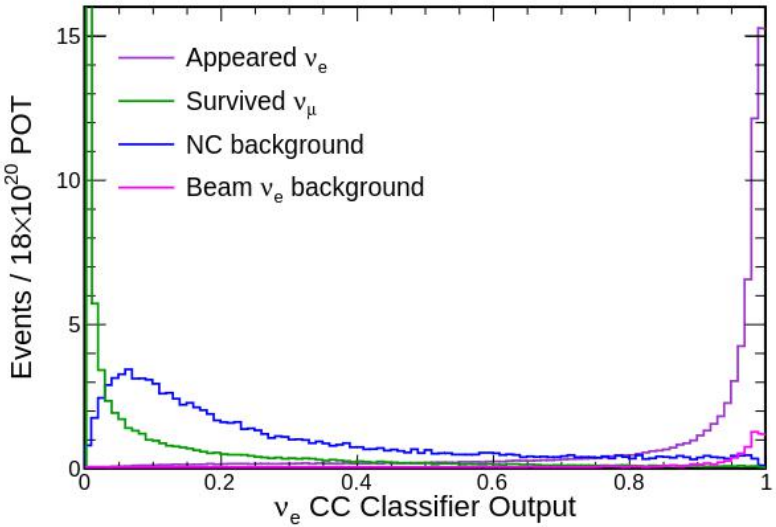


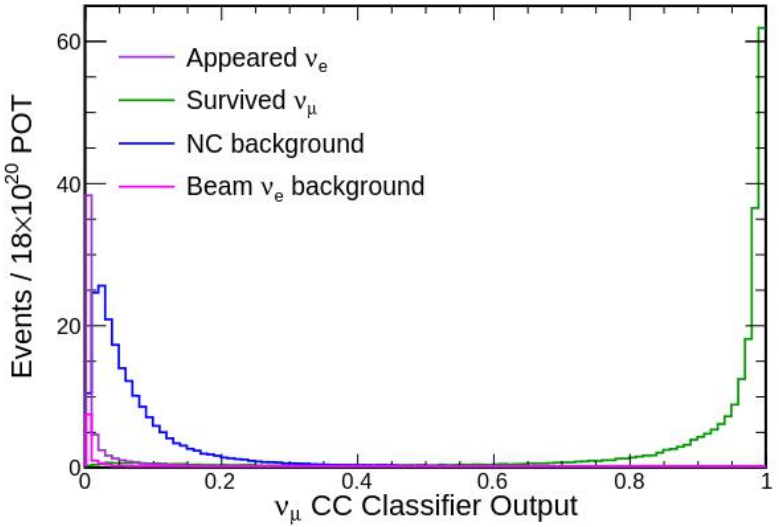


A Convolutional Neural Network Neutrino Event Classifier

CLASSIFICATION EXAMPLE: NOVA

Nova was one of the first experiments to really deploy convolutional neural networks into an experimental neutrino project.

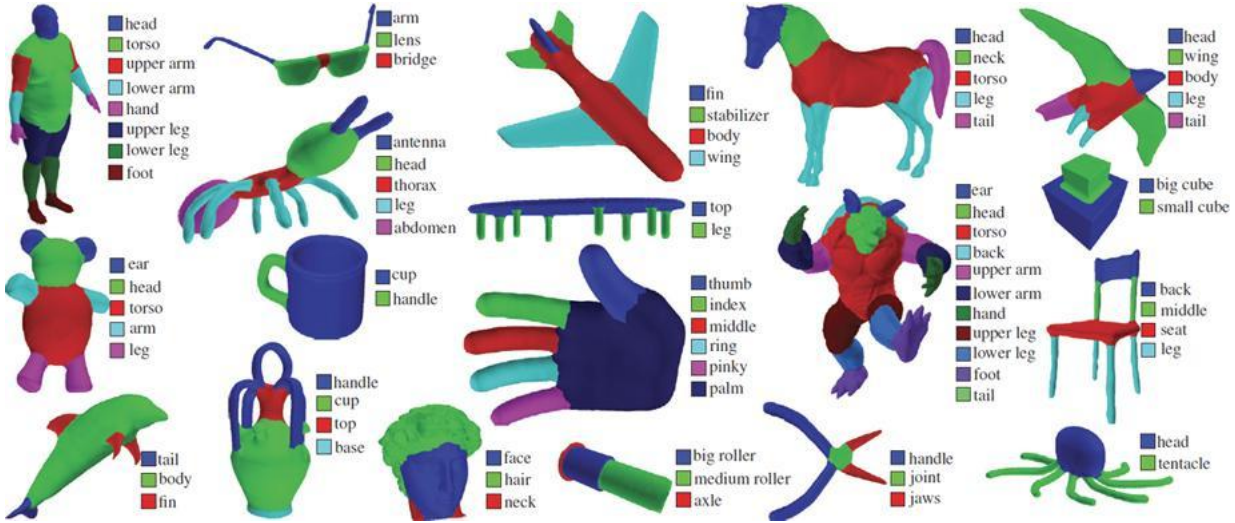




A Convolutional Neural Network Neutrino Event Classifier

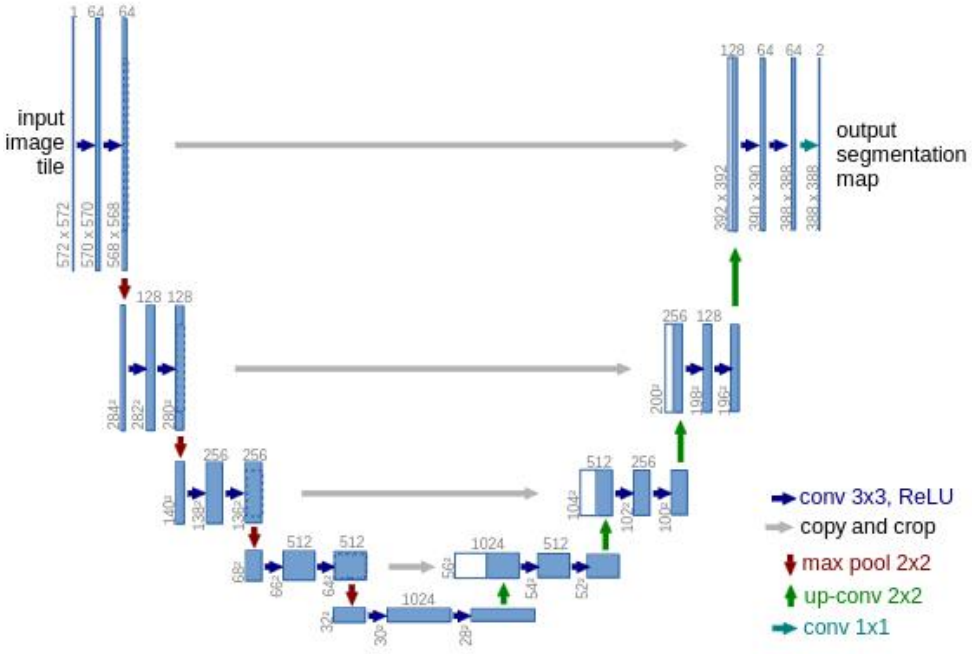
CNN SPECIALIZATION: SEGMENTATION

Instead of classifying the entire image, you can train the network to segment the image at the individual pixel level. No distinguishing between "instances" - each pixel just has a classification.



CNN SPECIALIZATION: SEGMENTATION

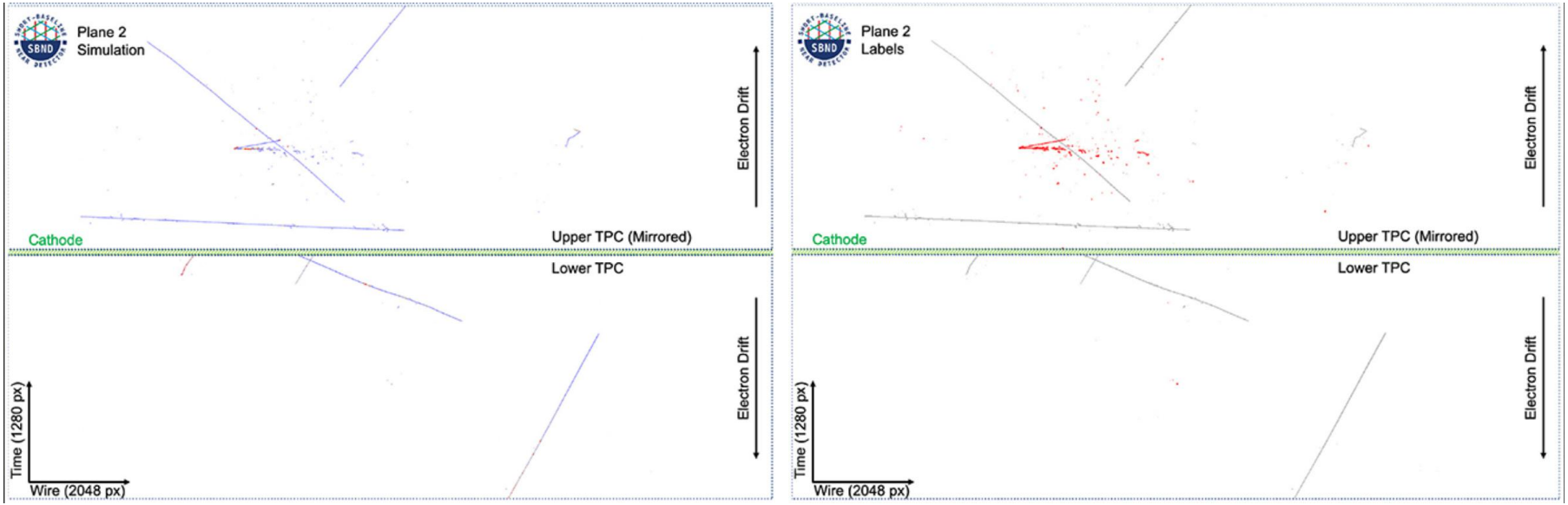
- UNet was (and still is!) a state of the art segmentation technique.
- Loss is calculated as a classification loss **per pixel.**
- Many applications in particle physics if they aren't **instance-specific.**
  - Background removal
  - Particle categorization (track/shower)
  - Region of Interest finding



SEGMENTATION CHALLENGES IN PHYSICS

- Our data is often more sparse than what has been used for development in CompSci.
  - "Every pixel is background" -> 99% accuracy in cosmic and neutrino tagging!
    - Finer grained metrics like "mean Intersection over Union" between truth and prediction sets of pixels can help distinguish performance between models.
  - Network can struggle to learn if the "rare" classes are also imbalanced.
    - Focal Loss (https://arxiv.org/abs/1708.02002v2) can really help with this.
- Segmentation tasks are often more computationally intense than classification, object detection.
  - Reduced precision can help accelerate training and inference, AND reduce GPU

SEGMENTATION EXAMPLE: COSMICTAGGER

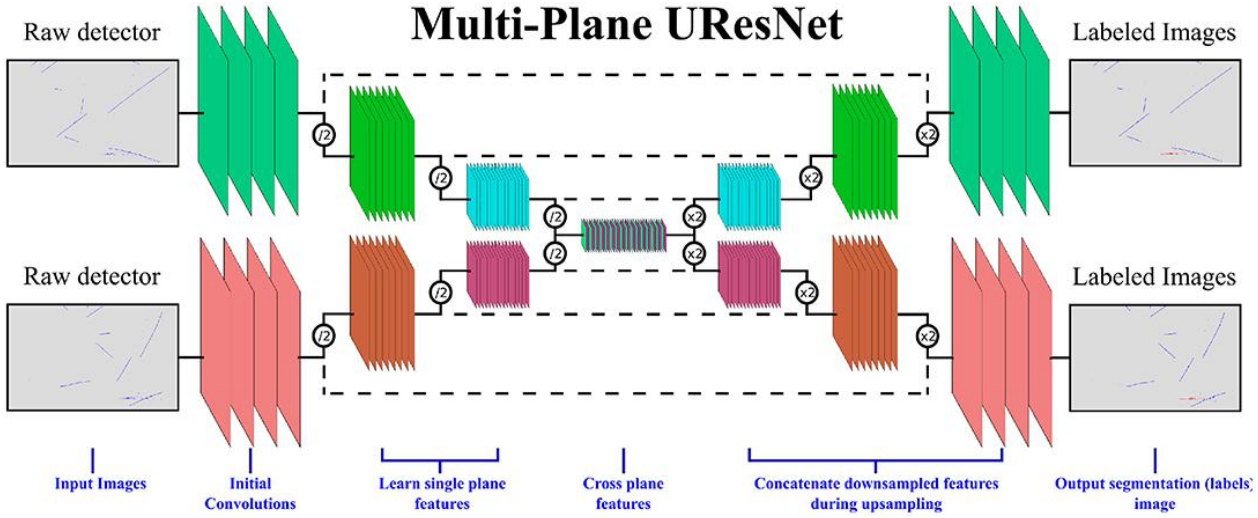


https://www.frontiersin.org/articles/10.3389/frai.2021.649917/full

Source code:
https://github.com/coreyjadams/CosmicTagger

SEGMENTATION EXAMPLE: COSMICTAGGER

- Modified UNet can learn from 3 planes at once and improve segmentation performance.
- Rejection power of cosmic-pixels 5x better than classical techniques.



https://www.frontiersin.org/articles/10.3389/frai.2021.649917/full

Source code:
https://github.com/coreyjadams/CosmicTagger

CNN SPECIALIZATION: GENERATIVE MODELS

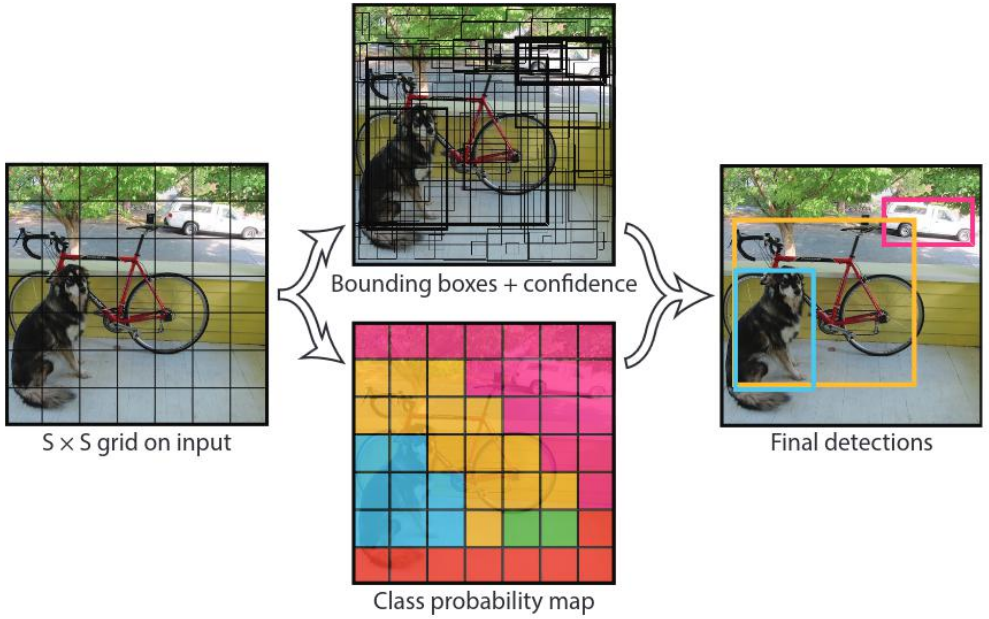
- You can use convolutional neural networks for generative models.
- “DCGAN” was one of the first really successfull GANs, now they are everywhere.
- We have an entire lecture on Simulation and Generative models in this school - I will leave it there. **BEWARE OF BIAS!!! (bias in == bias out ...)**

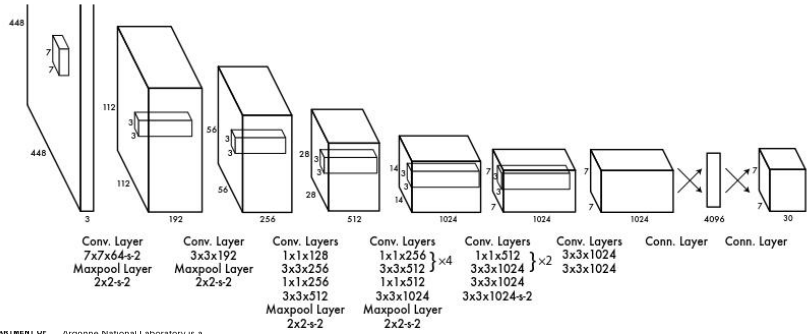


https://thispersondoesnotexist.com/

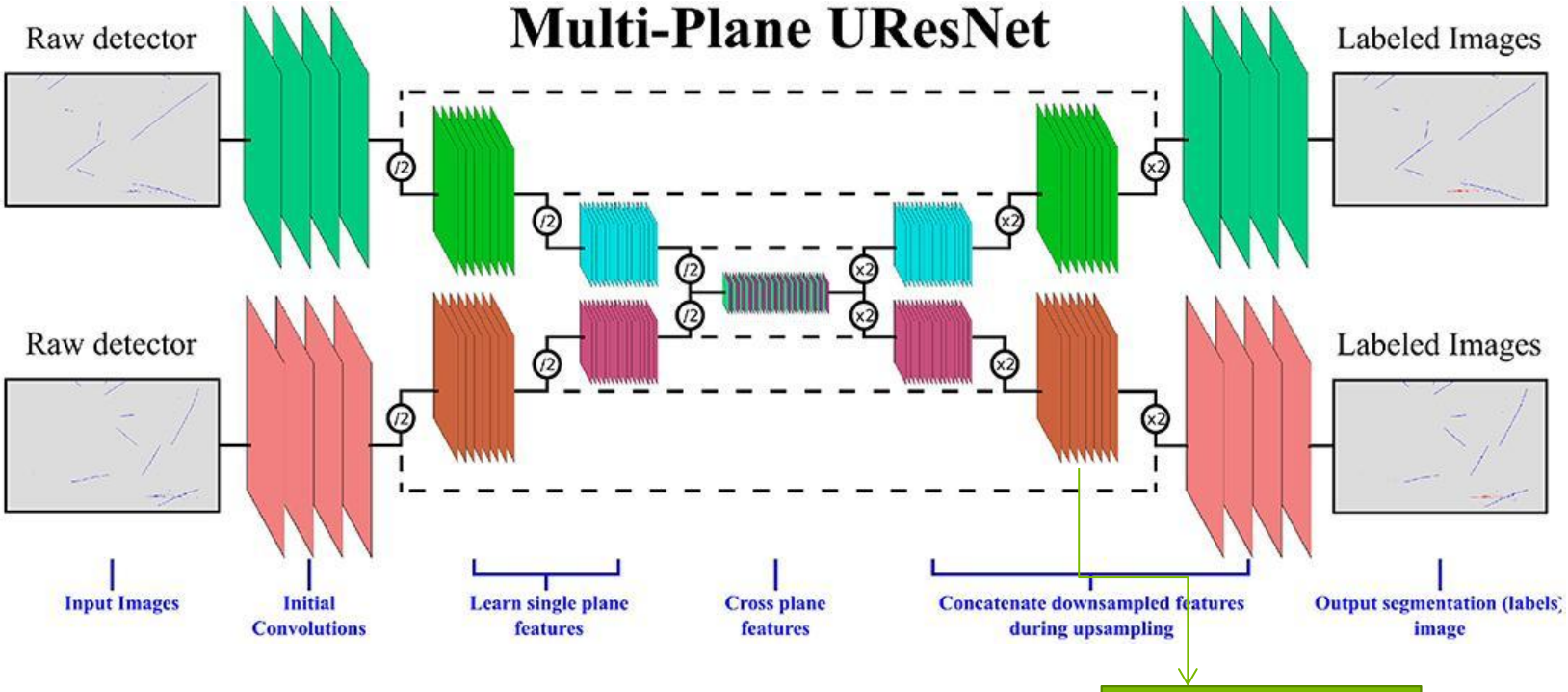
CNN SPECIALIZATION: INSTANCE DETECTION

Using the encoded features of a convolutional network, one can do instance-aware object detection (and subsequent segmentation)
YOLO (You Only Look Once)





CNN SPECIALIZATION: INSTANCE DETECTION



https://github.com/coreyjadams/CosmicTagger/tree/v2.1

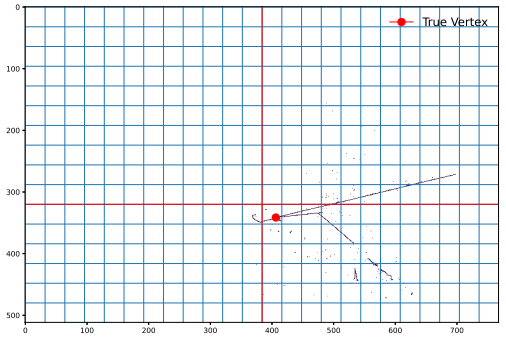
Vertex Proposal Network

CNN SPECIALIZATION: INSTANCE DETECTION

Output of the network is a 3-channel image, downsampled from original:

- Channel 0 is probability the vertex is in that box.
- Channels 1/2 are regression coordinates, relative location within a particlar **anchor box**

$$L = \sum_i FL(C_i, \widehat{C_i}) + \sum_i \mathbb{I}_i[(x_i - \widehat{x_i})^2 + (y_i - \widehat{y_i})^2]$$



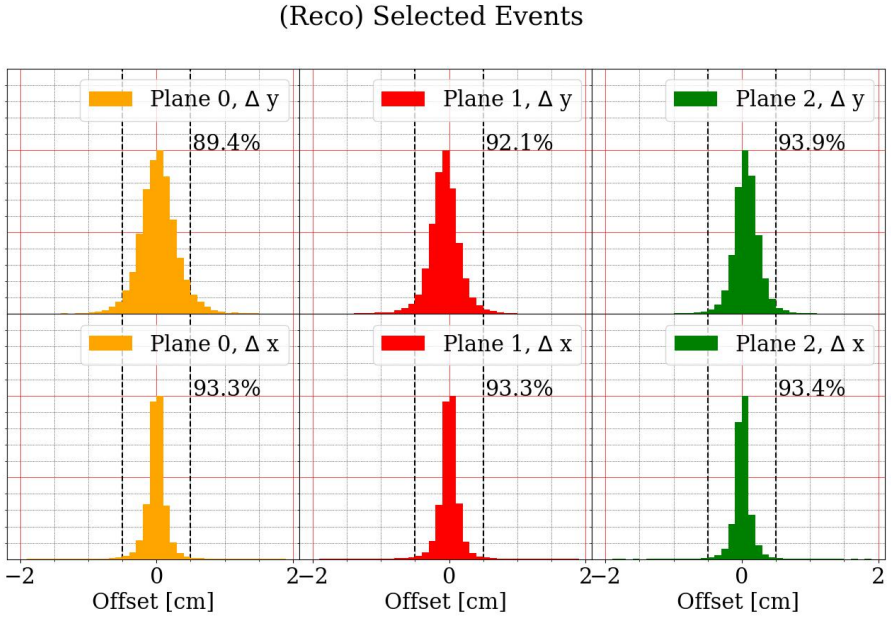
https://github.com/coreyjadams/CosmicTagger/tree/v2.1

CNN SPECIALIZATION: INSTANCE DETECTION

Output of the network is a 3-channel image, downsampled from original:

- Channel 0 is probability the vertex is in that box.
- Channels 1/2 are regression coordinates, relative location within a particlar **anchor box**

Vertex ID with a neural network is surprisingly easy! Want multiple points? Gets harder ...



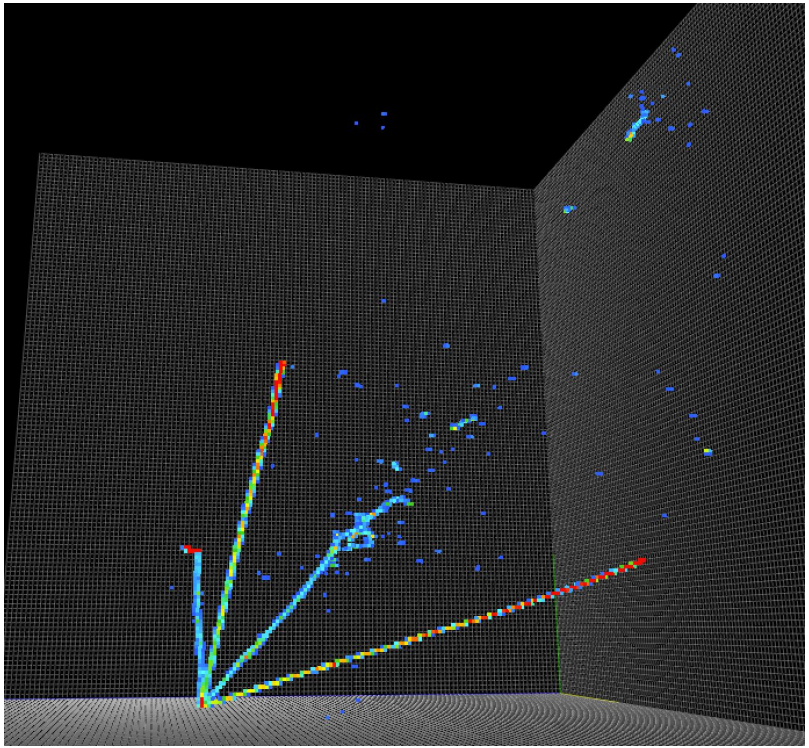
https://github.com/coreyjadams/CosmicTagger/tree/v2.1

HOW TO ADAPT A CNN TO PHYSICS

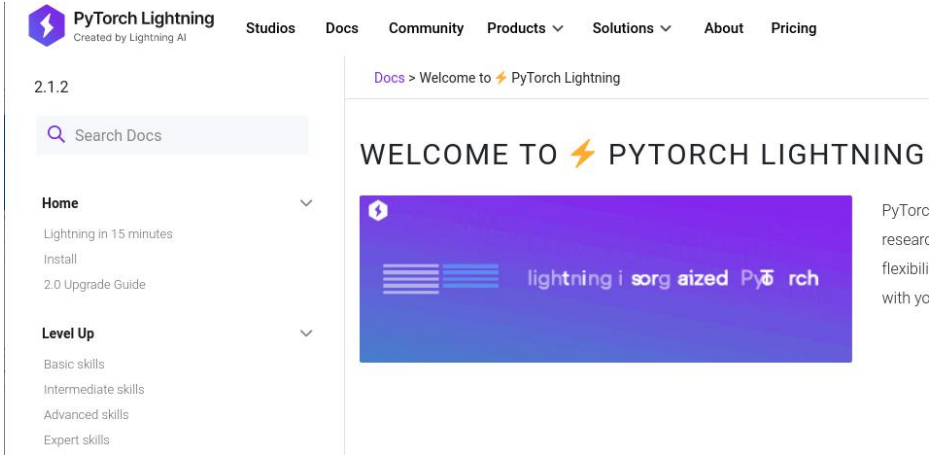
DATA IS EVERYTHING

- If you spend 90% of the project making sure the training data (sim?):
  - is correctly labeled (if possible),
  - represents the target data (detector data?),
  - is *enough* (how much is enough?),
  - has a train/val/test spilt,
  - is *balanced,*

Then you are doing it right!!



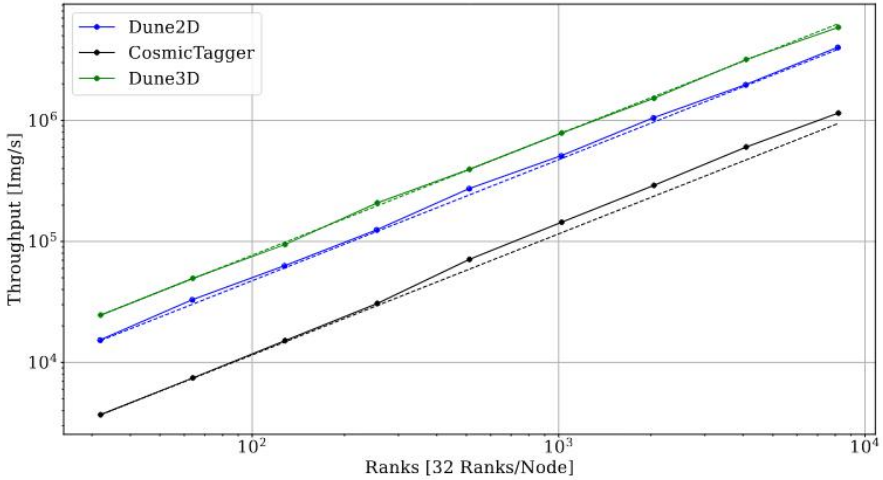
DON'T REINVENT WHEELS

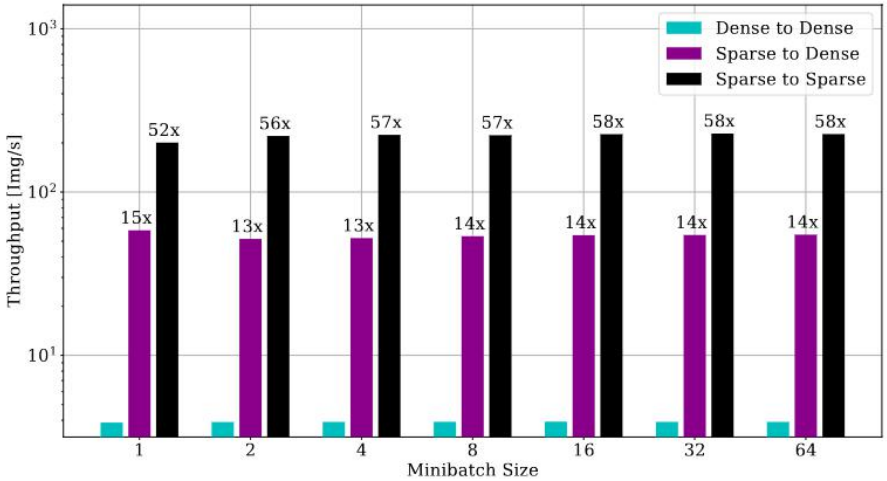


Fork repos and modify what you need - and at the same time: don't be afraid to dig deep in the code.

Focus on the science!
- Use GPUs to not waste your time training models.
- Take small samples of your data set to make sure your model and overfit and has the capacity to learn.
- It's *really easy* to use multiple GPUs these days!

IO SHOULDN'T STAND IN YOUR WAY





Sparse IO can be particularly powerful for particle data but the most important thing is to make sure it's not getting in your way!

https://arxiv.org/pdf/2209.04023.pdf

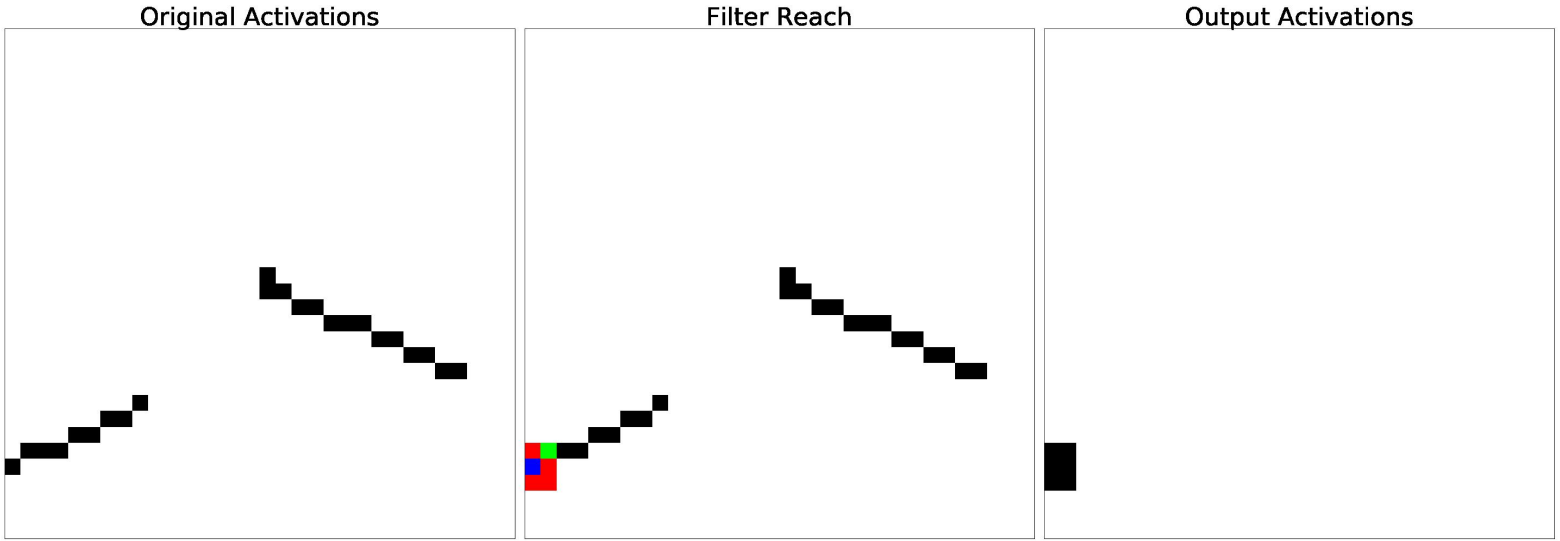
ABOUT COMPUTATIONAL PERFORMANCE

- AlexNet took O(1 week) to train initially. The current record for ResNet is 224 **seconds.**
- You don't have to be satisfied with long wait times to train your models!!
- GPU computing is available - free for you scientific needs! - at several national laboratories: ALCF@ANL, OLCF@ORNL, NERSC@LBNL
- Think your code is slow/fast already? It is worth it to **profile.** Check out the **line_profiler** package for python.
- Many models require batch sizes that are larger than can fit on a single GPU. Check out **data parallel learning** (horovod, DDP, DeepSpeed, and more) to increase your batch size without slowing down your training (much).

TORCH, TF, JAX?

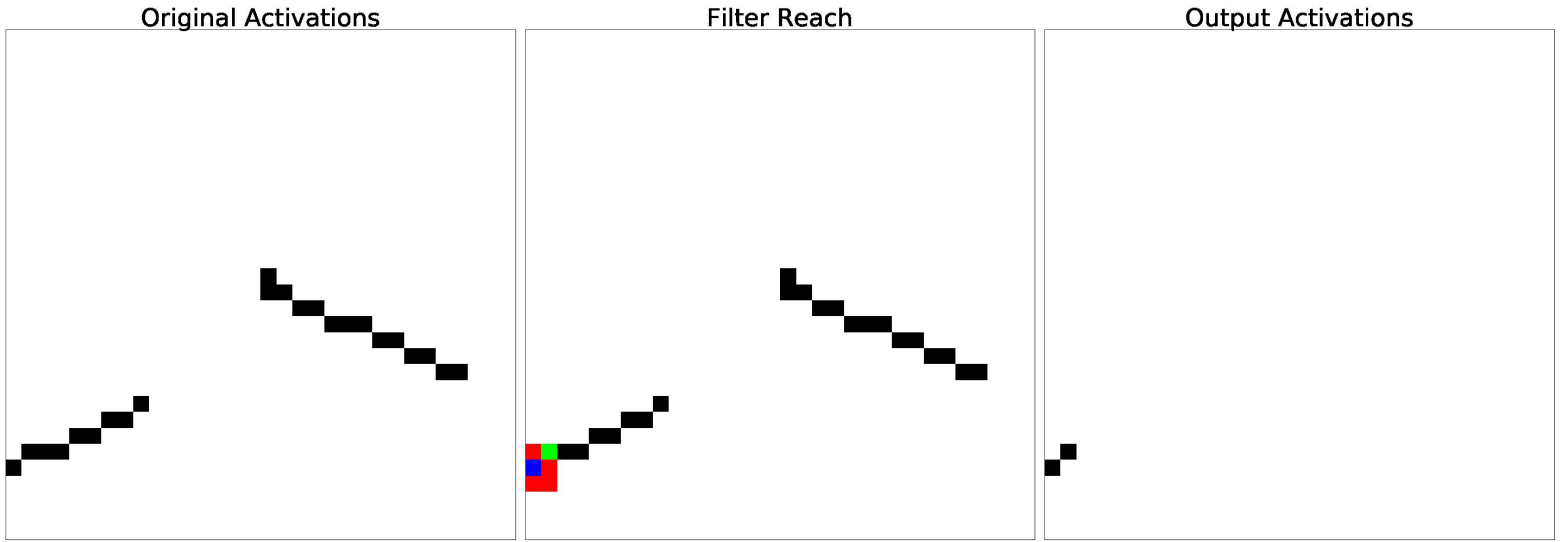
- For convolutional neural networks, use whatever is easiest for you!
  - Got a local guru in one of these frameworks? Use what they use.
  - Does your collaboration have infrastructure to use one already? Use that?
- No constraints? Pytorch is often the simplest and easiest to use.
- JAX is the newest framework and due to it's JIT compiler, it is probably the fastest - IF you write your code right!
- For GNNs: pytorch_geometric is the industry standard (JAX's Jraph might also be good)
  - Much more on GNNs in my second lecture...

SPARSE CONVOLUTIONAL NETWORKS



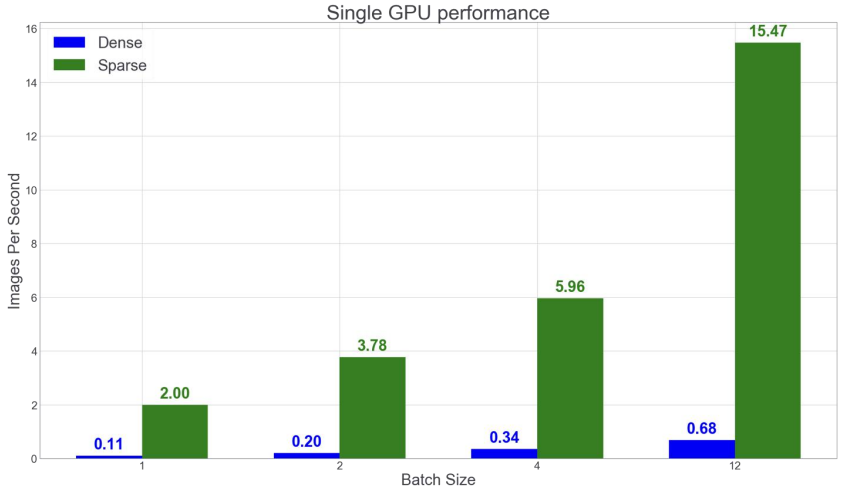
Sparse Convolution looks only at non-0 input pixels

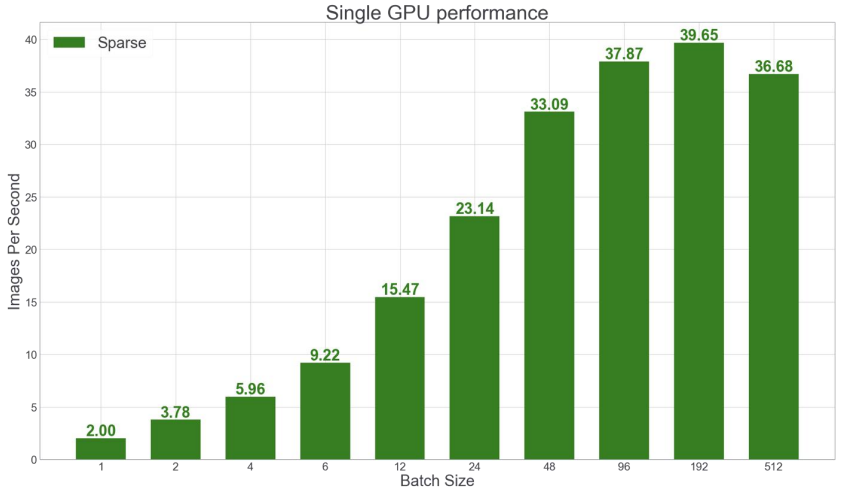
SPARSE CONVOLUTIONAL NETWORKS



Submanifold Sparse Convolution looks only at non-0 input pixels

SPARSE IS FASTER THAN DENSE FOR PARTICLE DATA

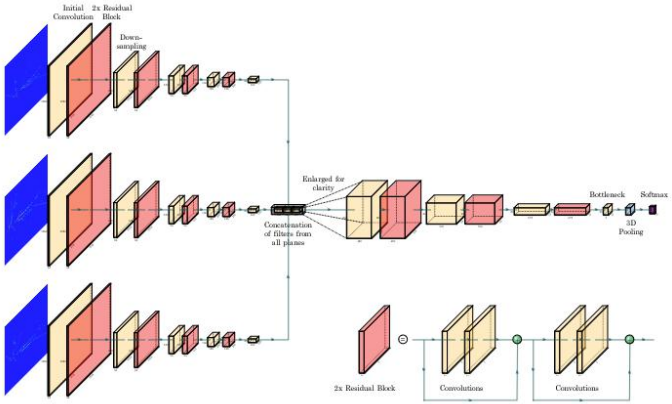


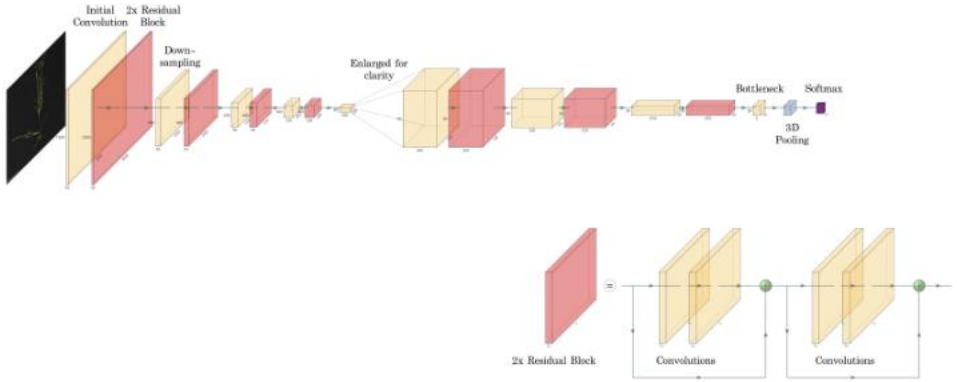


Roughly 18x speedup in training on identical batch sizes, 58x speedup in peak single-GPU throughput.

Performance gain can depend on sparsity of your data.
Even better throughput possiblem by oversubscribing the GPU with multiple processes (but more tricky to tune).

SPARSE CONVOLUTIONS ENABLE 3D CNNS





3D CNNs have **huge** memory requirements: high resolution data requires high GPU memory during training.

Sparsity improves in 3D, though!

Category	Accuracy [%] 3D	2D
Neutrino Interaction	**94**	91
Proton Multiplicity	**91**	87
Charge Pion Presence	**94**	91
Neutral Pion Presence	**95**	94

https://arxiv.org/pdf/1912.10133.pdf
https://github.com/coreyjadams/SparseEventID

WRAPPING UP CNNS

- CNNs are extremely powerful tools for image analysis.
- Valuable for classification, segmentation, instance-aware predictions.
- Not too hard to apply to physics!
  - The computer scientists have done excellent work building models and architectures.
  - There is plenty of room for physicists to make impacts in computer science - but don't hesitate to focus on the physics either it that's your goal!
- Sparse convolutional neural networks can be perfect for sparse particle data.
  - Particularly for 3D!
- Next time:
  - Mostly GNNs
  - Will start with a little bit of "training on sim vs. running on data"