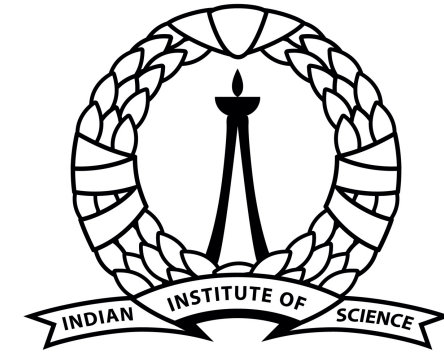




Department of Computational and Data Sciences



भारतीय विज्ञान संस्थान

Computer Vision

Deepak Subramani

Assistant Professor

Dept. of Computational and Data Science

Indian Institute of Science Bengaluru

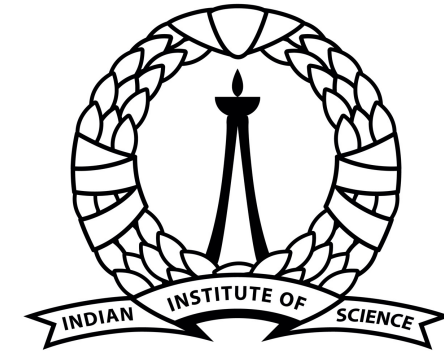


Lecture and Assignment Guide

- This Slide Deck has Material for 6 hours of teaching divided into Parts 1-6
- We will go through
 - Week 01
 - Part 01 - Convolutional and Pooling Layers; AST 01
 - Part 02 - Transfer Learning and Modern CV Design Principle; AST 02
 - Week 02
 - Part 01 - Modern Convolutional Building Blocks for Image Classification; AST 03
 - Part 02 - Object Localization
 - Interpreting what convolutions learn (Advanced topic) – AST 03
 - Week 03
 - Part 01 - Object Detection (YOLO), Image Segmentation – Lec 05
 - Part 02 - Practical CVOps
 - AST04 – Object Detection with YOLO
 - Week 04
 - Revision
 - AST05 – Image Segmentation
- Additional Reading material to go in depth of math with references and code references are provided with the marking of “Additional Material” or “Additional Discussion” etc



Department of Computational and Data Sciences



भारतीय विज्ञान संस्थान

CV Week 02: Part 01

Deepak Subramani

Assistant Professor

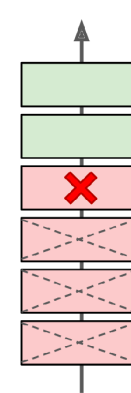
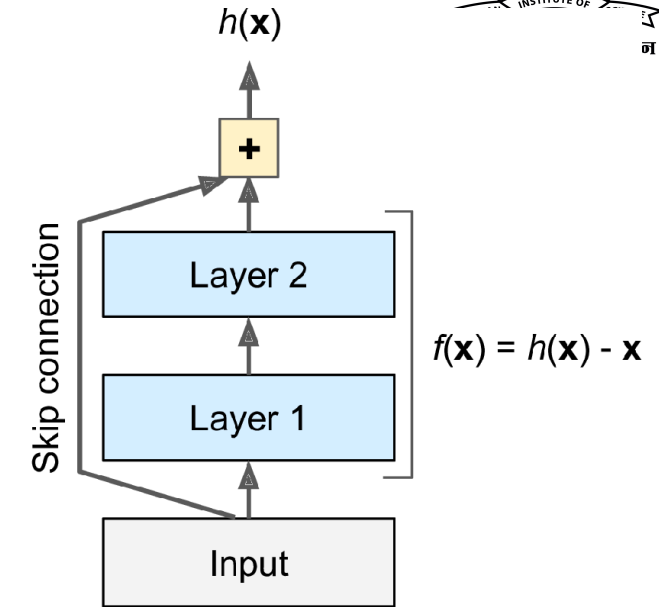
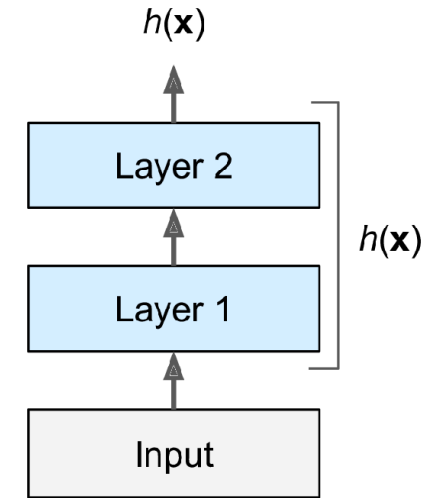
Dept. of Computational and Data Science



Indian Institute of Science Bengaluru

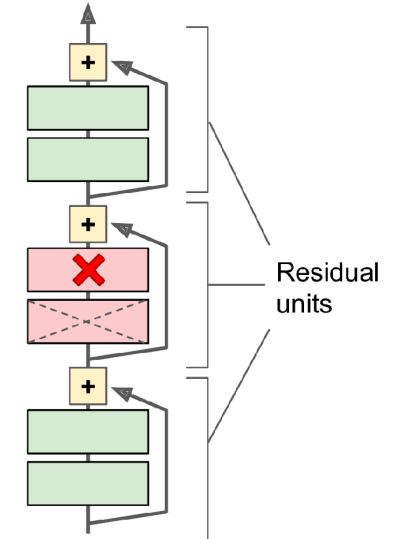


Residual Block

- 2015 winner is ResNet that used a residual block
- Networks were being deeper and residual (or skip connections) enabled training such deeper networks
- Usually networks are trained to learn a function $h(x)$
- By adding a skip connection, we are forcing the network to learn $f(x) = h(x) - x$
- When stacking several Residual Units, the signal can make its way to all the parts of network even if some layers experience a vanishing gradient



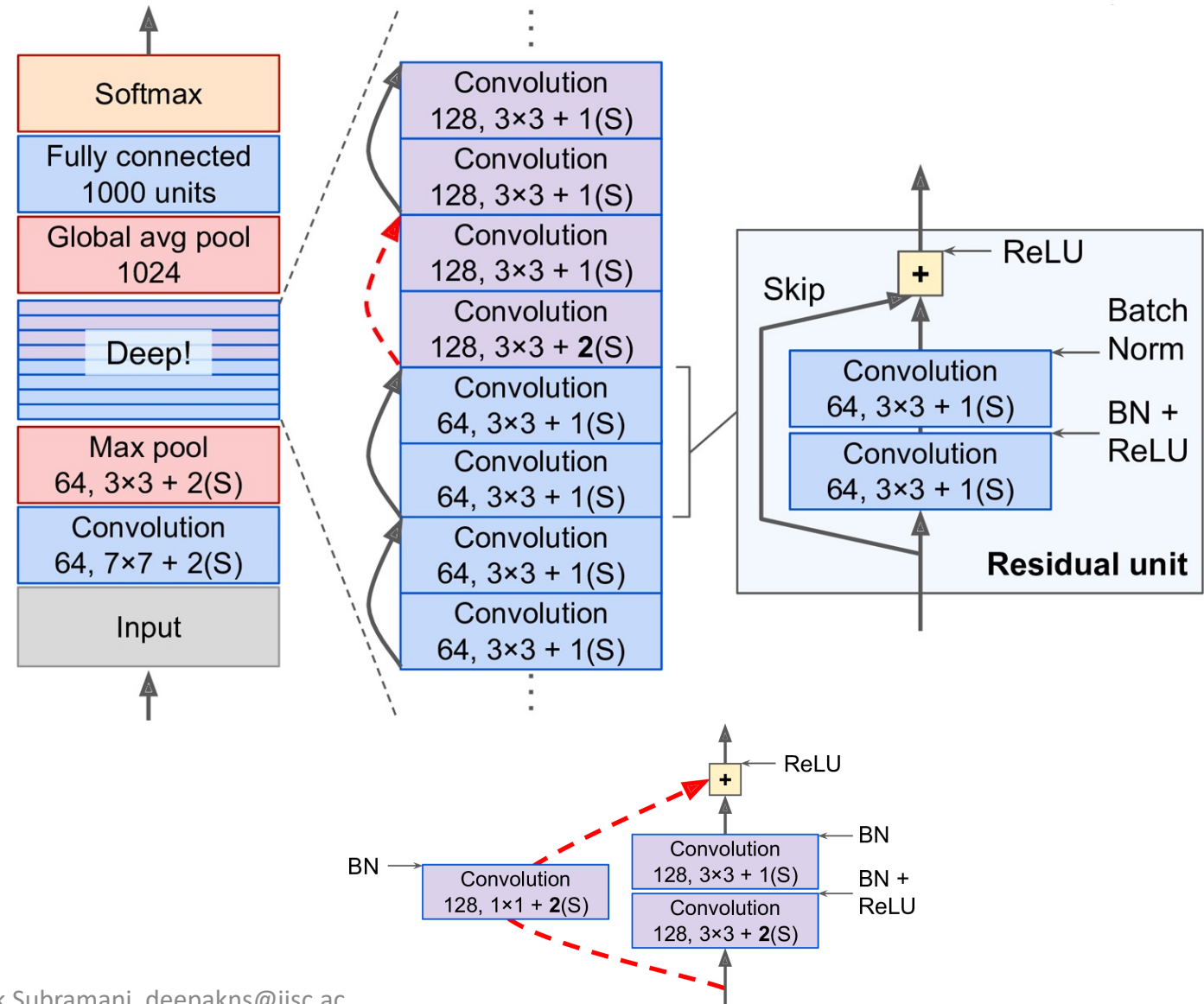
 = Layer blocking backpropagation
 = Layer not learning





ResNet

- The Resnet looks and feels like the GoogLeNet
- But uses RU instead of Inception Modules
- In a RU, 2 conv layers are used
 - After 1st BN+ReLU
 - After Second apply BN, Skip and then ReLU
- After a few RU, the number of filters are doubles and ht,wd are halved
 - A direct skip is not possible here



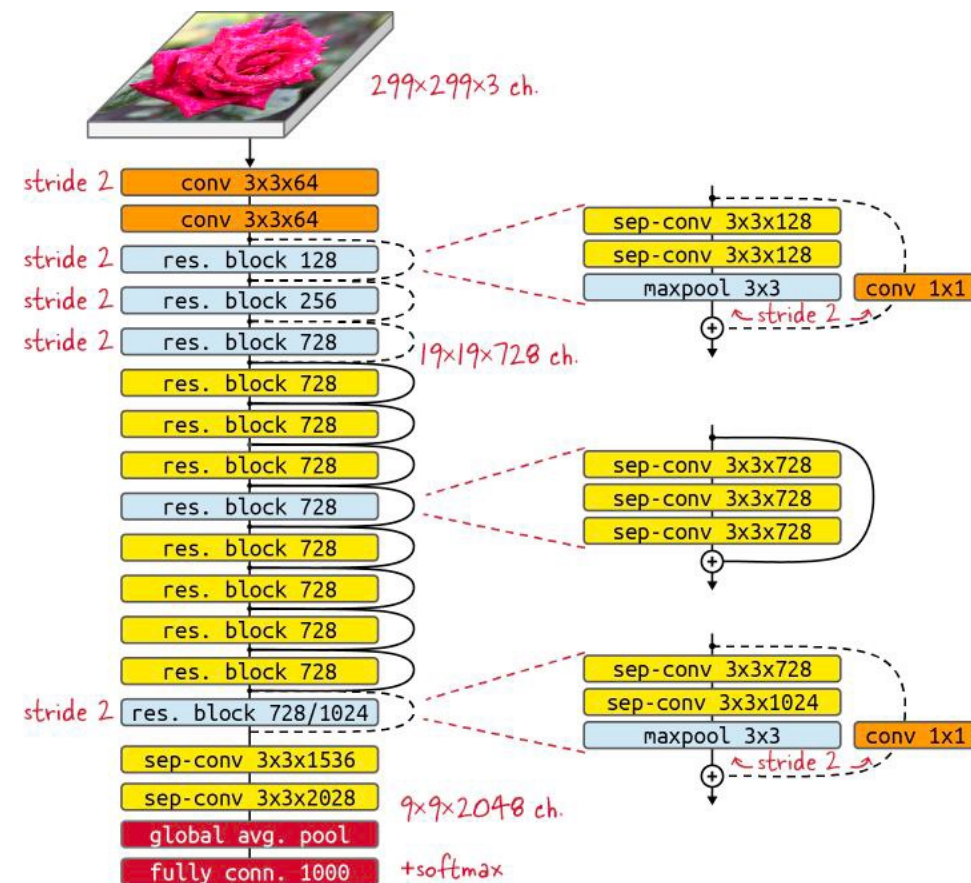
Batch Normalization

- He initialization + ELU can reduce vanishing/exploding gradient problem at the beginning, but problems can recur later during training
- Batch Normalization (Ioffe and Szegedy 2015) solves this problem
- Idea:
 - Zero center and normalize before or after activation function of every layer
 - Learn two parameter vectors (one set for every input) – output scaling and output shift – i.e., learn the optimal mean and scale of each of the layer's inputs!
- Question:
 - Need a batch to calculate the mean and std for scaling
 - Use the current mini batch to get the mean and std
- Note:
 - Add BN after input layer, then it is almost equivalent to applying StandardScaler, but only on the mini-batch and not the full train set



Xception

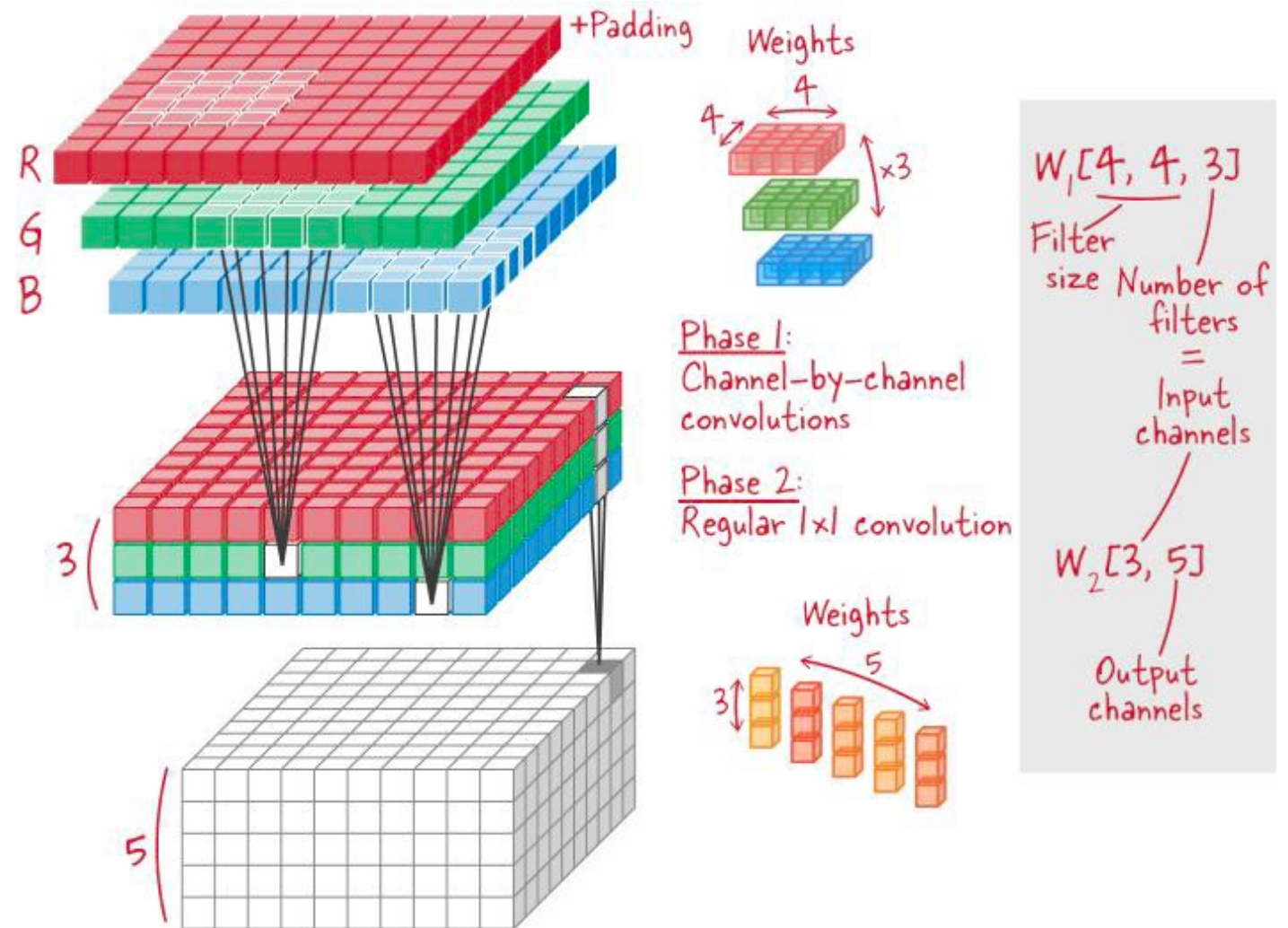
- Xception was proposed in 2016 by Francois Chollet
- It is a variant of the GoogLeNet
- Introduced depth wise separable convolutional layer
- 2 parts
 - 1 applies spatial filters only one per input channel
 - 2 applies 1x1 depth convolution only
- 1 looks exclusively for spatial patterns and 2 for depth
- Similarity to GoogLeNet
 - 1x1 and reg conv net
 - Here 1x1 and only spatial





Depthwise Separable Convolutions

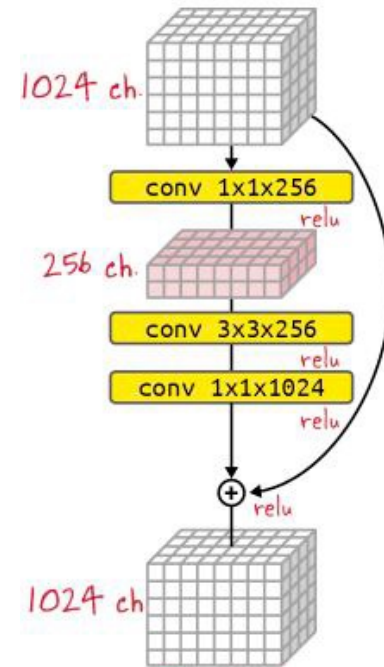
- Channel-by-channel convolutions followed by 1x1 Conv



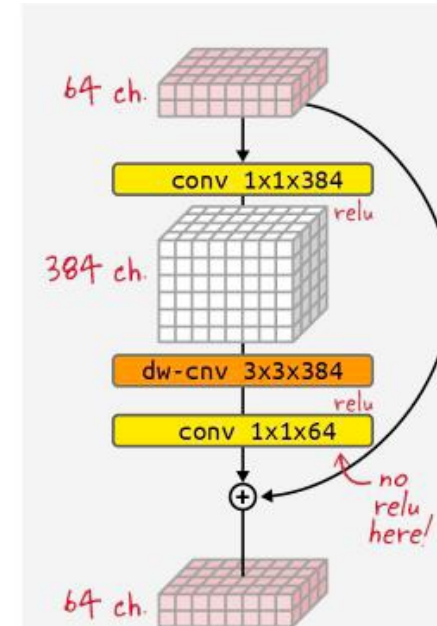


Inverted Residual Bottleneck

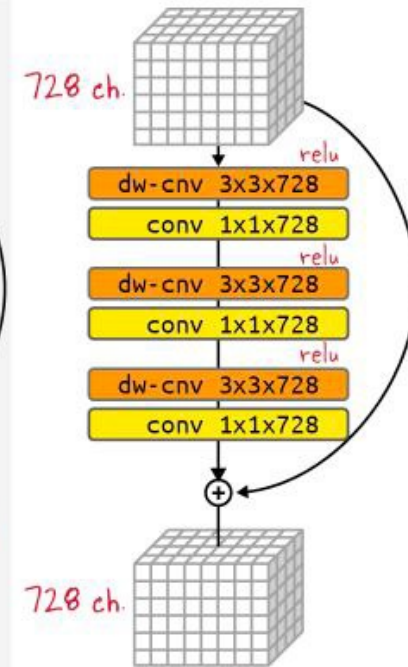
- Goal: Same expressivity as ResNet, Xception but with a dramatically reduced weight count and inference time
- Designed to be used on mobile phone where resources are scarce
- Argument: Information flow between residual blocks is low-dim in nature and can be represented by limited number of channels
- Important: Last 1x1 doesn't have any nonlinear activation as ReLU would destroy too much information



ResNet
"many - few - many"
1.1M channels, weights



Inverted Residual Bottleneck
"few - many - few"
channels, 52K weights

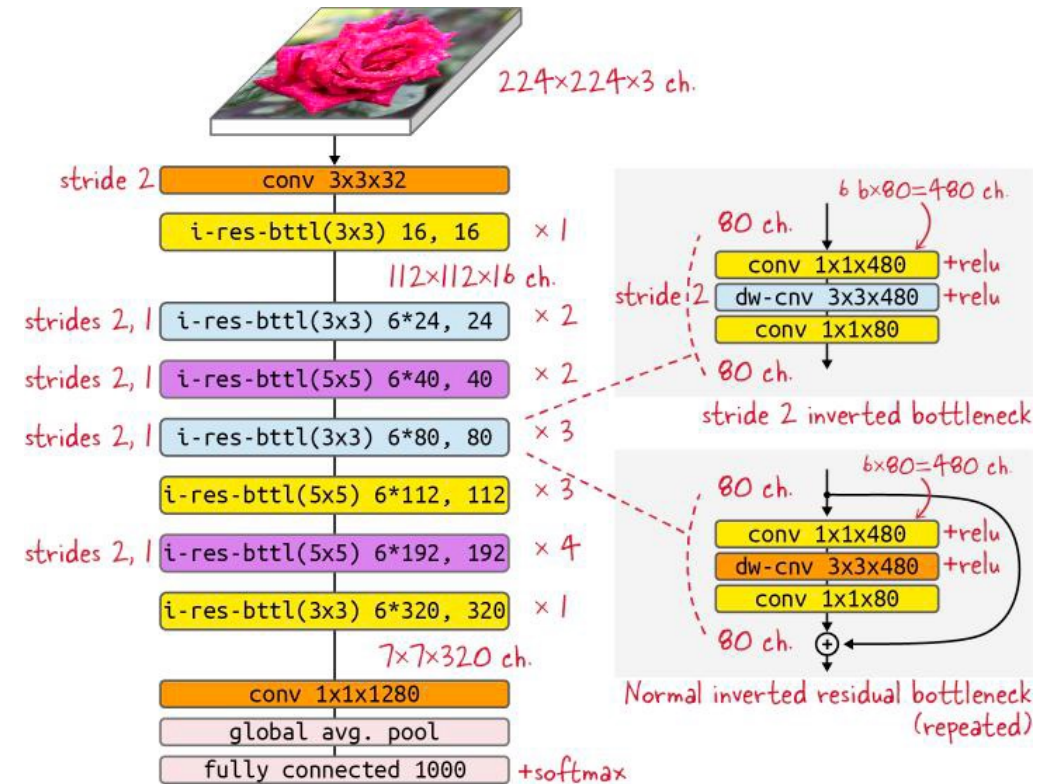


Xception
"many - many - many"
channels, 1.6M weights



EfficientNet

- Neural Architecture Search – A principled search to find Neural Networks that beat human designed networks
- Efficient Net is the state of the art Convolutional Neural Network (CVPR 2019) for ImageNet Challenge
 - Later winners and current SoTA are Vision Transformer-Based Models
- Set of models whose numbers in the [i-res-bttl(K,K) P*Ch, Ch]xN changes depending on the available compute



What to read of all these architectures?

- Architecture: Which layers, how to configure and what arrangement
- Architecture defines the hypothesis space of the model
 - The set of possible functions that can be learnt by the SGD
- Like feature engineering, building a good hypothesis space includes our prior knowledge of the task at hand
 - Eg: Using conv nets, we know that our patterns are translation invariant
- Remember that Gradient Descent is stupid! – It needs all the help we can provide
- A good architecture reduces the search space and promotes convergence
- Architecture is more art than science
 - Experience helps you cobble together powerful learners, beginners struggle to get a network to train!
 - Keyword is “intuition”, and following some best practices
 - The special structures we saw are the best practices



Importance of ablation study in research

- Deep learning architectures are evolved rather than designed—
 - repeated trying and selecting what seems to work.
 - Much like in biological systems, if you take any complicated experimental deep learning setup, chances are you can remove a few modules (or replace some trained features with random ones) with no loss of performance.
- Things are made worse by the competitive publishing culture
 - Incentives are for making a system more complex, thereby signalling that it is more interesting and more novel
 - If you read many DL papers, you will see that they are optimized for peer review than to communicate an interesting finding
- Do Ablation Studies: “Could there be a simpler explanation? Is this added complexity really necessary? Why?”
- In Ablation study, let us say you got a model with $X+Y+Z$ to work, see what X , Y , Z , $X+Y$, $Y+Z$, $X+Z$ produce and compare them



How much data is needed for ConvNet training?

- Exact number for “enough” is relative – it depends on the number of parameters to be learnt
- Fortunately, convnets learn local translation invariant features
- Thus, they are in fact highly data efficient for perceptual tasks
- Small, well regularized convnets for simple tasks (e.g., binary classification) can be trained with small image datasets
- Further, for Computer Vision tasks, many pre-trained models are available that can be reused

Data Augmentation

- Having too few examples can cause overfitting
- When only limited data is available, an artificial Data Augmentation can be performed to produce believable looking images
- Some Data Augmentation techniques include
 - Random Flipping
 - Random Rotation
 - Random Zoom
- In keras, data_augmentation layers can be added right after the input
 - These layers are active only during training and have no effect during inference
- They allow the model to learn features without memorization



Interpreting what Conv Nets learn

- There is always a feeling that deep networks are not interpretable
- When writing papers and convincing stakeholders it is important to show what the deep learning network learnt
- For convnets, since most applications involve using spatial (image) data, three different techniques can be used for interpretation
 - Visualizing intermediate activations
 - Visualizing convnet filters
 - Visualizing heatmaps of class activation

Thinking about Layers

- Convolutional layers learn feature maps
- These feature maps are translation invariant features present in different parts of the image
- The bottleneck layers extract the low dimensional representation of the feature space
- Coupled conv layers are single layers on steroids
- Skip connection does residual learning
- Depthwise convolution learns features across channels
- Lower layers get broad general features, and upper layers learn finer details for the task
 - Sometimes these learned features are not interpretable



Department of Computational and Data Sciences

Additional Material



- BN Math and Code
- AlexNet
- GoogleNet
- MobileNet
- SENet

BN Algorithm: Operations at a BN Layer

1. $\boldsymbol{\mu}_B = \frac{1}{m_B} \sum_{i=1}^{m_B} \mathbf{x}^{(i)}$ Calculate the mean of every of the inputs
2. $\sigma_B^2 = \frac{1}{m_B} \sum_{i=1}^{m_B} [\mathbf{x}^{(i)} - \boldsymbol{\mu}_B]^2$ Calculate variance
3. $\hat{\mathbf{x}}^{(i)} = \frac{\mathbf{x}^{(i)} - \boldsymbol{\mu}_B}{\sqrt{\sigma_B^2 + \epsilon}}$ Remove mean and divide by std; ϵ is a small number to avoid division by zero
4. $\mathbf{z}^{(i)} = \boldsymbol{\gamma} \otimes \hat{\mathbf{x}}^{(i)} + \boldsymbol{\beta}$ Scale by $\boldsymbol{\gamma}$ and shift by $\boldsymbol{\beta}$, which are trainable parameters

\otimes is element wise multiplication [Hadamard Product]

$\mathbf{z}^{(i)}$ is the output from the BN layer

Behavior during Testing

- During training we have a mini batch, so the mean and variance can be calculated
- But what about testing?
- Things are more difficult as we may sometimes test only on one input and not a mini-batch of inputs – then what?
- Solution
 - Wait until training is completed, then calculate a mean and variance for the entire data and use it during testing
 - Use a moving average of the inputs mean and standard deviation for use in the testing - Keras does this automatically
 - So 4 parameters are learnt by Keras
 - γ, β using back prop. μ, σ^2 using exponential moving average
 - μ, σ^2 are used only during testing and not during training

Advantages of Batch Normalization

- Batch Normalization improves the accuracy of almost all neural networks
- Vanishing gradient problem was strongly reduced
 - With BN, even saturating activation functions such as tanh and sigmoid can be used
- Networks were less sensitive to weight initialization
- Larger learning rates can be used that speed up training
 - Each epoch is slower with BN, but fewer epochs only are needed
- BN also acts like a regularizer
- Disadvantage: It adds more complexity and adds to the runtime during inference as well
 - But the BN weights can be fused to the previous layer thereby avoiding the runtime penalty
 - TFLite does this automatically

BN Code

```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(300, activation="elu", kernel_initializer="he_normal"),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(100, activation="elu", kernel_initializer="he_normal"),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(10, activation="softmax")
])
```

```
>>> model.summary()
Model: "sequential_3"
```

Layer (type)	Output Shape	Param #
flatten_3 (Flatten)	(None, 784)	0
batch_normalization_v2 (Batch Normalization)	(None, 784)	3136
dense_50 (Dense)	(None, 300)	235500
batch_normalization_v2_1 (Batch Normalization)	(None, 300)	1200
dense_51 (Dense)	(None, 100)	30100
batch_normalization_v2_2 (Batch Normalization)	(None, 100)	400
dense_52 (Dense)	(None, 10)	1010
Total params: 271,346		
Trainable params: 268,978		
Non-trainable params: 2,368		

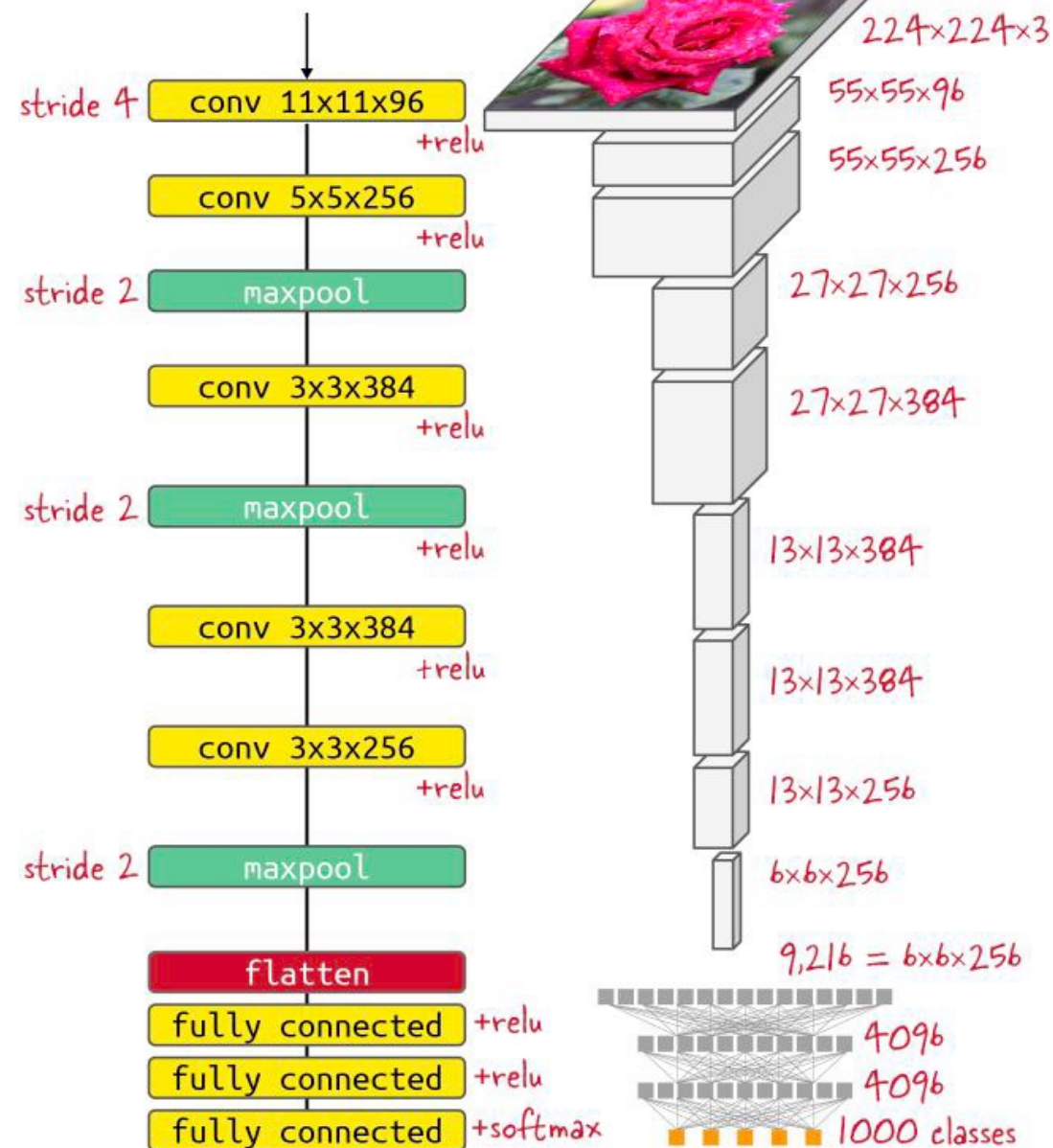
μ, σ are the non-trainable parameters
 Total BN params = 3136+1200+400 = 4736
 Half of this (4736/2 = 2368) is non-trainable
 Trainable is w.r.t. backprop

Although the original paper stated that batch normalization operates by “reducing internal covariate shift,” no one really knows for sure why batch normalization helps. There are various hypotheses, but no certitudes. You’ll find that this is true of many things in deep learning—deep learning is not an exact science, but a set of ever-changing, empirically derived engineering best practices, woven together by unreliable narratives. You will sometimes feel like the book you have in hand tells you *how* to do something but doesn’t quite satisfactorily say *why* it works: that’s because we know the how but we don’t know the why. Whenever a reliable explanation is available, I make sure to mention it. Batch normalization isn’t one of those cases.



Alex Net

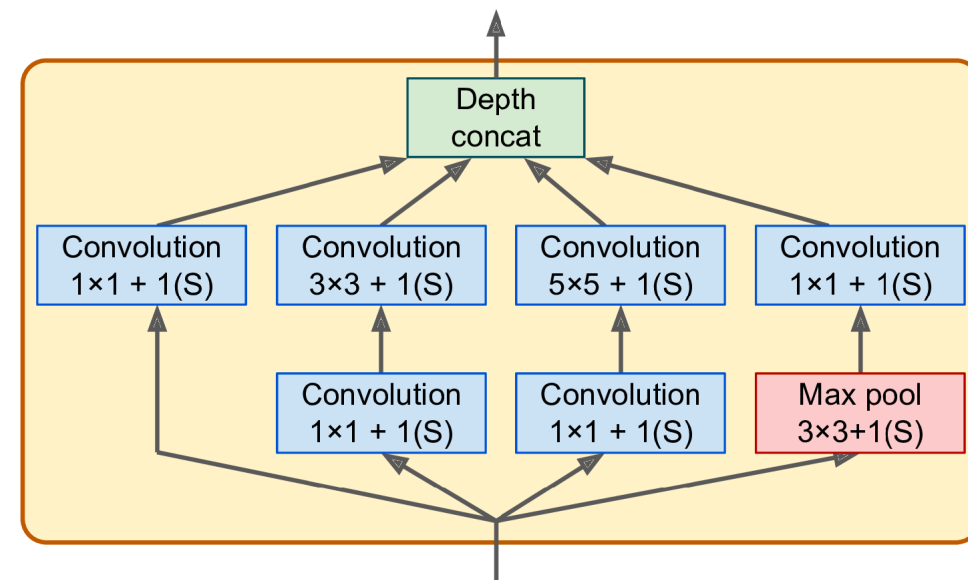
- Winner of 2012 ImageNet Challenge
- Larger and deeper ConvNet
- First time ConvNets were stacked on top of one another
- Innovations:
 - Data Augmentation
 - Random flipping, increase brightness, rotation
 - Local Response Normalization





Inception Module

- $3 \times 3 + 1(S)$ means use a 3×3 kernel, with stride 1 and “same” padding
- Input is copied to four different layers
- The output of three are fed to a second convolutional layers
- Finally, the output of all the 4 convolutional layers are concatenated along the depth direction
- All convolutional layers use ReLU
- All layers use stride=1 and same padding, so all input and output sizes are same
- They also act as bottleneck layers, reducing parameters and improving generalization



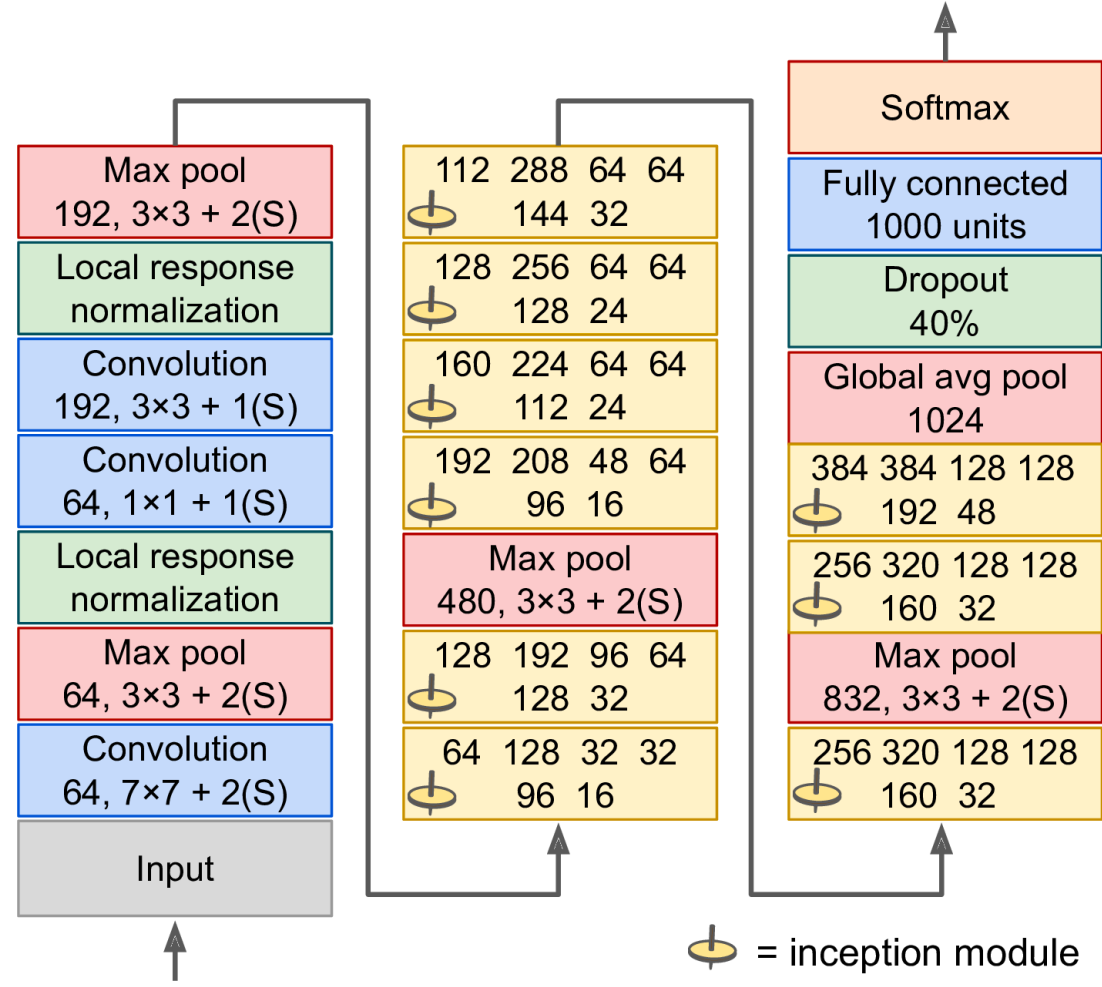
Geron Fig 14-13

Each pair of convolutional layers ($[1 \times 1, 3 \times 3]$ and $[1 \times 1, 5 \times 5]$) acts like a single powerful convolutional layer, capable of capturing more complex patterns.



GoogLeNet

- First two layers use stride=2 each and reduce the image size by 4 in ht and wd direction (16 overall)
- LRN makes the previous layer learn a wide variety of features
- Pair of 1x1, 3x3 layers (like the inception) act as a smart conv layer. 1x1 layer is an info bottleneck
- LRN makes prev layers capture wide variety of features
- 9 inception modules with 2 maxpool layers to reduce dimensionality
- Depth of each part of inception layer is mentioned
- After inception (so far 5 layers with stride=2), the image size is down by 32x32.
 - Input is 224x224, so now it is 7x7
- Last inception layer returns 1024 channels
- Global avg pool returns the average of each channel
 - All spatial info is now lost, anyway we are doing a classification
- Dropout and 1000 nodes for the 1000-class classification

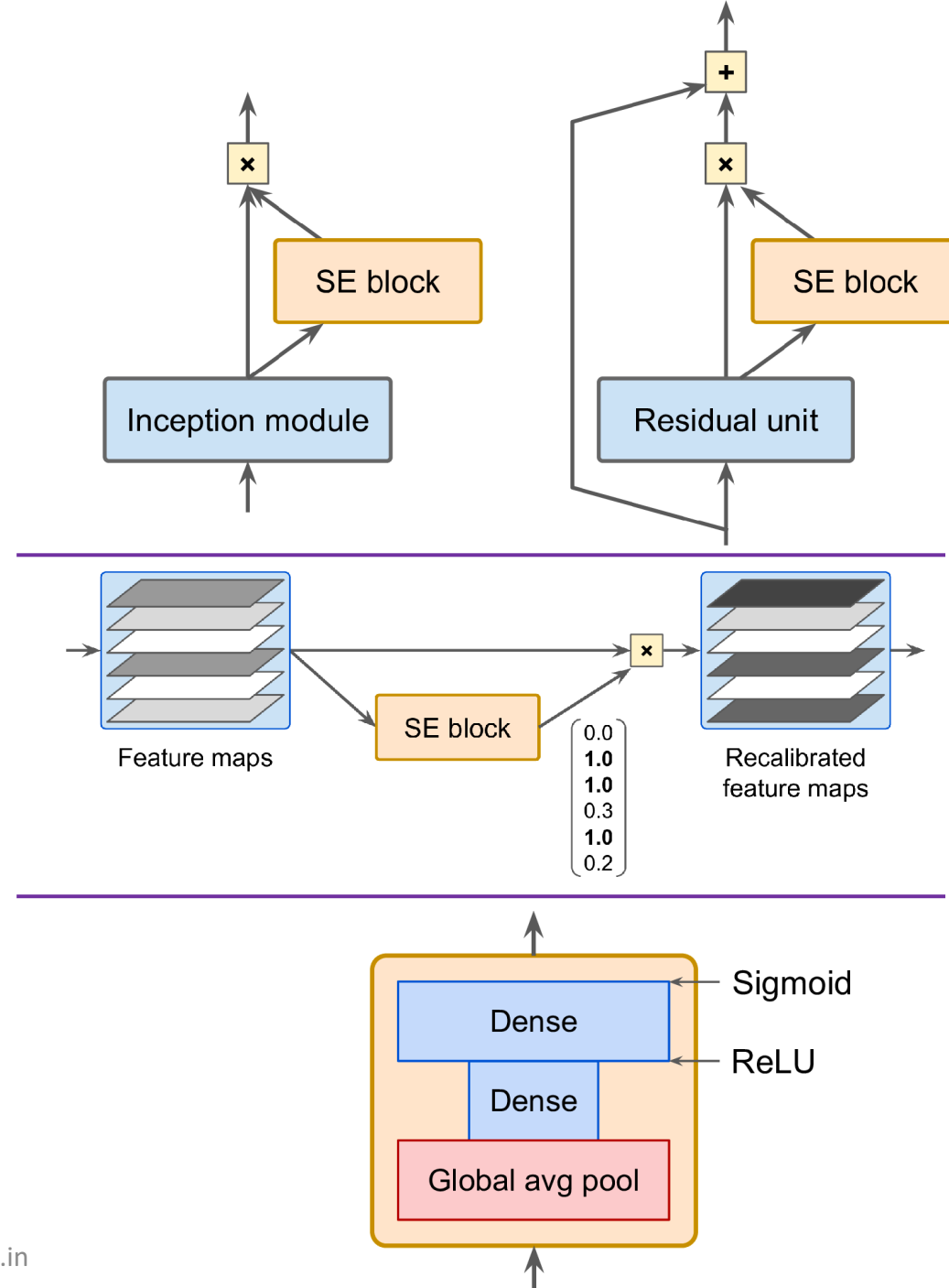


Geron Fig 14-14



SENet

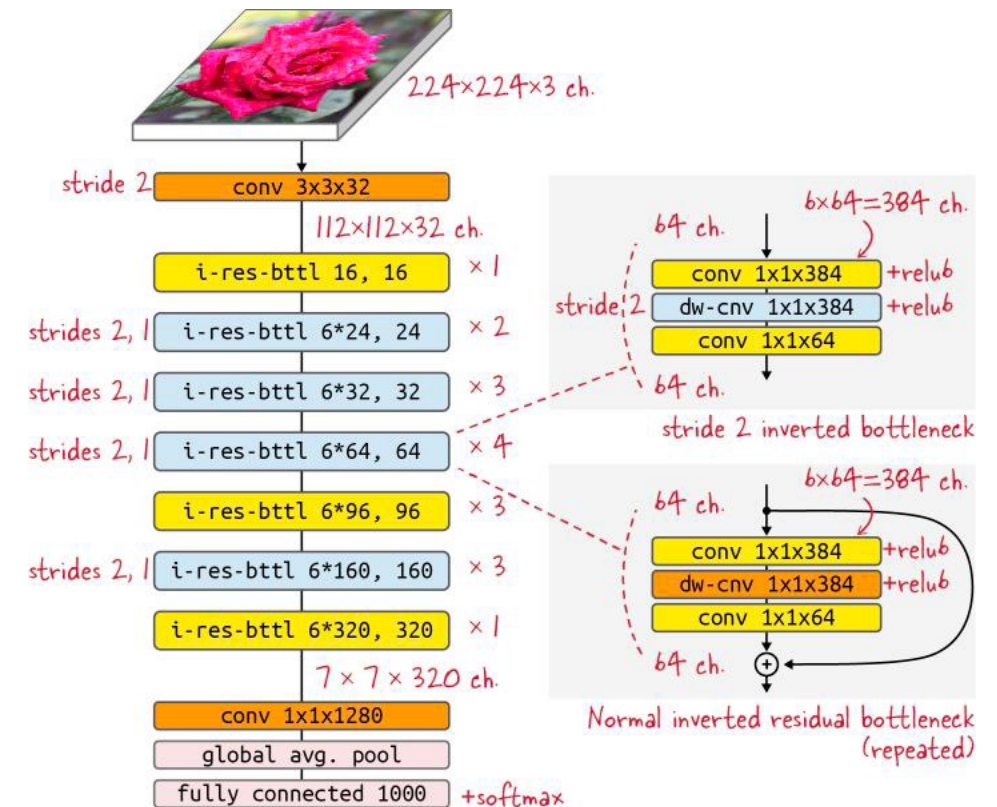
- Winner of 2017 Image Net is Squeeze and Excitation Net (SENet)
- Improve RU and Inception module by adding an SE block
- SE block working:
 - analyzes the output of the unit it is attached
 - Focuses only on the depth dimension
 - Learns which features occur together
 - Recalibrate the feature map with this information
- SE block architecture
 - Global avg pooling to output a single number per feature map
 - Dense is the squeeze layer that contains less neurons than the number of feature maps in the previous layer (finds an embedding)
 - Output dense layer produces as many numbers as the feature maps
 - This is used to recalibrate the feature maps by multiplication





MobileNet V2

- “i-res-bttl N, M” is parameterized by their internal (N) and external channel depth (M).
- Every sequence marked “strides 2, 1” starts with an inverted bottleneck block with a stride of 2 and no skip connection. The sequence continues with regular inverted residual blocks.
- All convolutional layers use batch normalization.





Department of Computational and Data Sciences

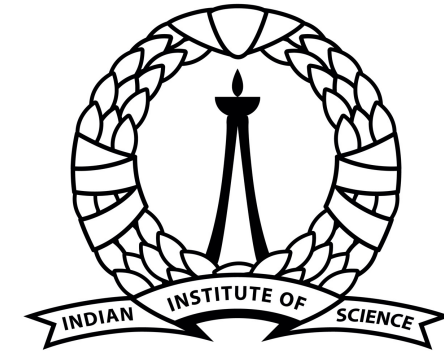
End of Additional Material



- BN Math and Code
- AlexNet
- GoogleNet
- MobileNet
- SENet



Department of Computational and Data Sciences



भारतीय विज्ञान संस्थान

CV Week 02 Part 02

Deepak Subramani

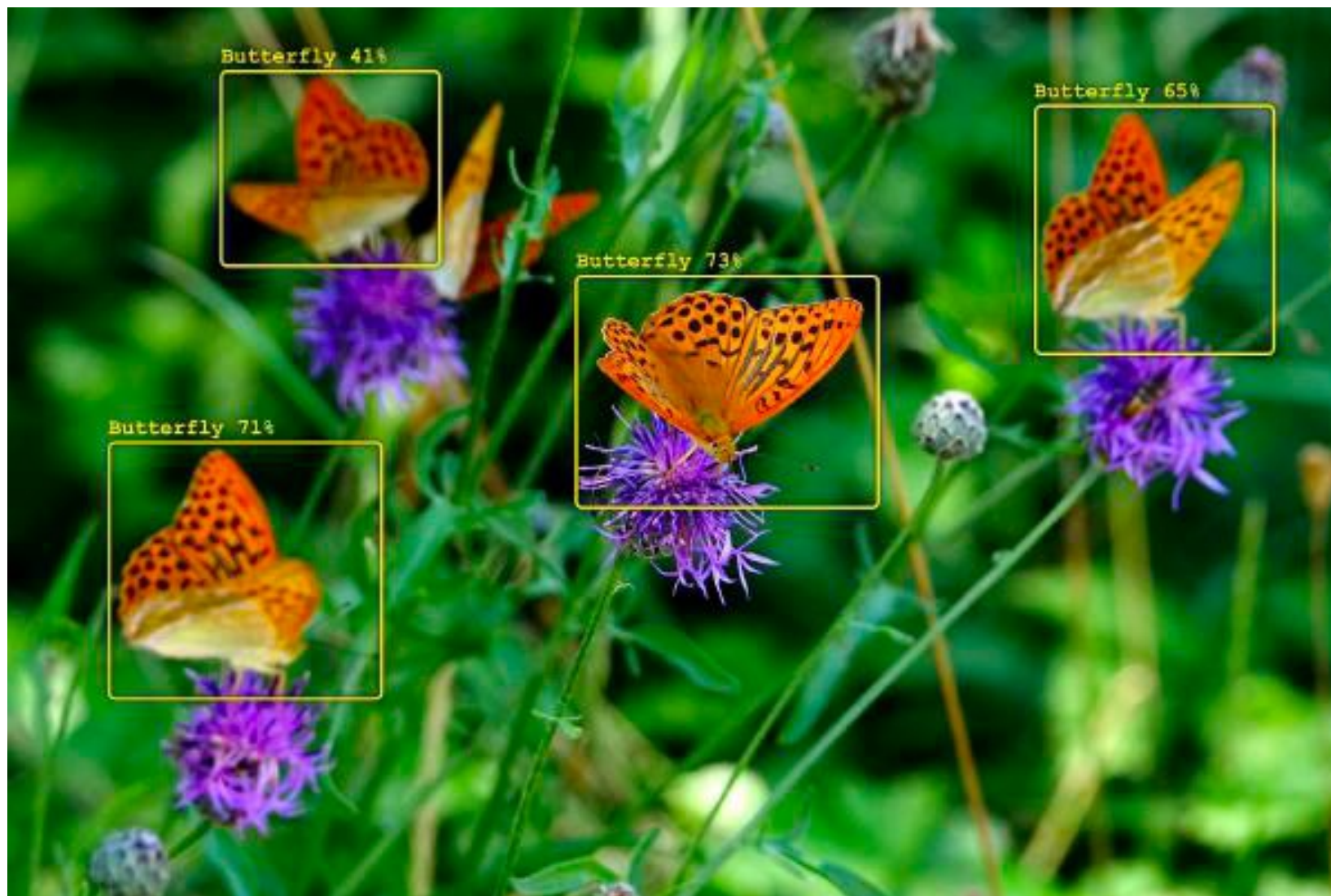
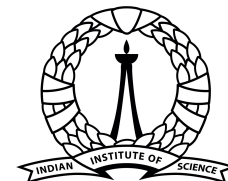
Assistant Professor

Dept. of Computational and Data Science

Indian Institute of Science Bengaluru



Introduction





Object Localization

- Object localization is a regression task
 - Predict a bounding box
 - 4 numbers
 - Horizontal and vertical coordinate of object center
 - Height and width of image

```
base_model = keras.applications.xception.Xception(weights="imagenet",  
                                                    include_top=False)  
avg = keras.layers.GlobalAveragePooling2D()(base_model.output)  
class_output = keras.layers.Dense(n_classes, activation="softmax")(avg)  
loc_output = keras.layers.Dense(4)(avg)  
model = keras.Model(inputs=base_model.input,  
                     outputs=[class_output, loc_output])  
model.compile(loss=["sparse_categorical_crossentropy", "mse"],  
              loss_weights=[0.8, 0.2], # depends on what you care most about  
              optimizer=optimizer, metrics=["accuracy"])
```

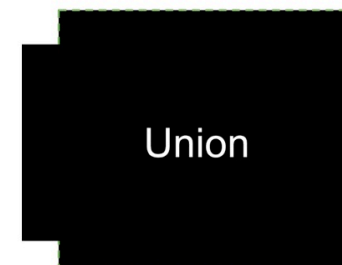
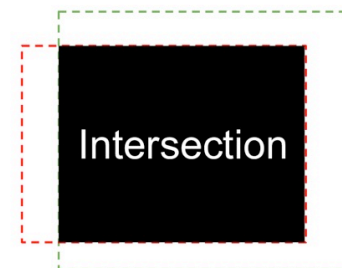
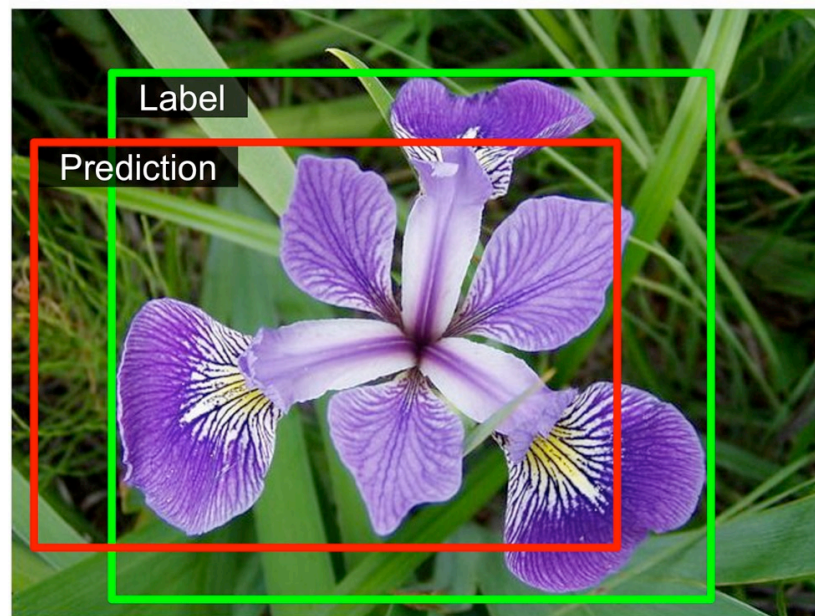
Object Localization: Data

- The label problem
 - To train we need bounding boxes in the training data
 - Getting labels is hard and costly
 - You should use some open source image annotator (e.g., VGG Image Annotator) or a commercial tool [Google Search for annotators!]
 - Crowdsourcing is also possible, but expensive and a task in itself!
- Things to remember:
 - Bounding boxes are normalized so that all 4 numbers (2 coordinates, height, width) are all between 0 and 1
 - Commonly square root of height and width is predicted
 - Why?: 10-pixel error of a larger bounding box is penalized less than a 10-pixel error of a smaller bounding box



Loss Function and Evaluation for Localization

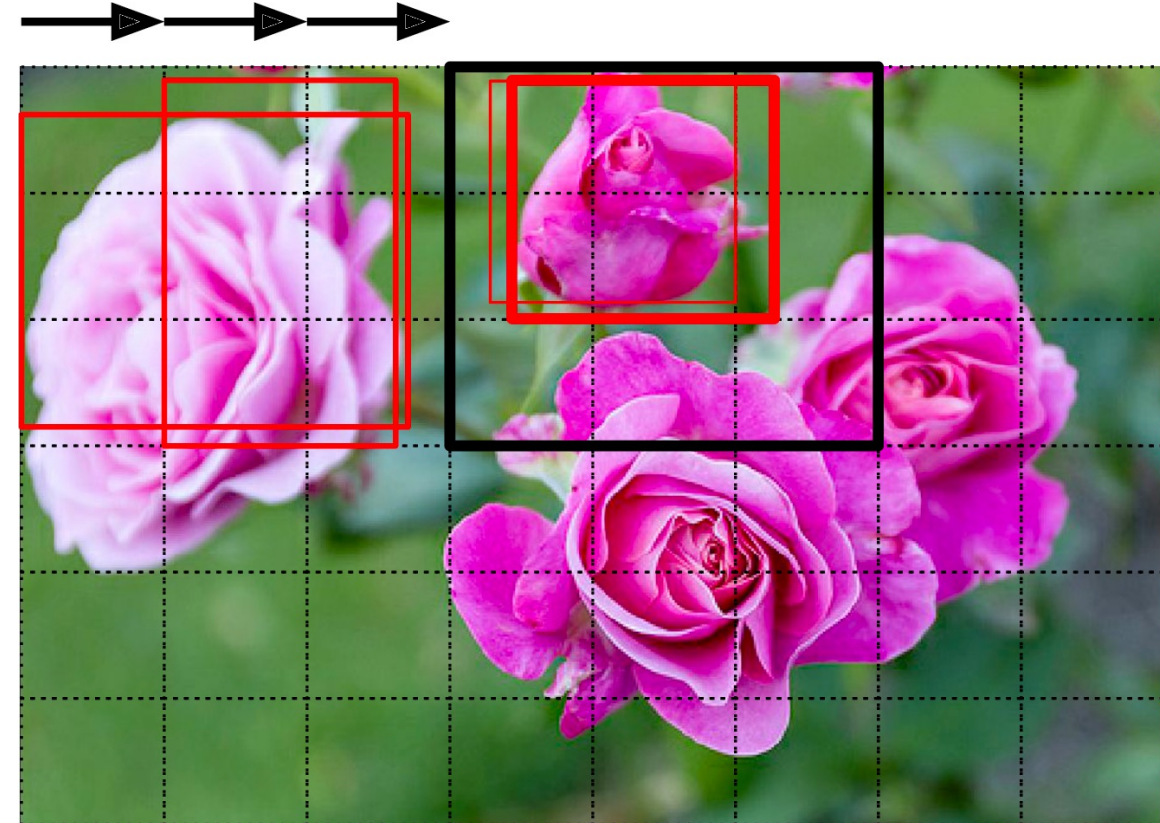
- MSE works well as a loss function,
 - but not a great metric to evaluate model performance
- Most common evaluation metric is Intersection over Union (IoU)
- IoU is not differentiable, so it is not used as a loss function
 - some approximations exist, but not practically useful or better than MSE Loss





Object Detection

- Object Detection: The task of simultaneously classifying and localizing multiple objects in an image
- Earlier approaches involved sliding a single object detector CNN over an image and use post processing
- Steps
 - Slide bounding box CNN over the image
 - Find bounding boxes
 - Post process



Object Detection

- Steps
 - Divide the image into grids
 - Slide an object detector over each 3x3 region
 - Predict one bounding box per 3x3 region
 - Repeat for 4x4 etc
 - In addition to a bounding box, output “objectness” – the probability of the object being present in the bounding box
 - Sigmoid activation trained with binary CE
 - Postprocess to get rid of overlapping bounding boxes
 - Drop BBox with low objectness
 - Find and drop BBox that overlap with the BBox with highest objectness
 - Repeat until no more boxes can be removed

