Activation functions

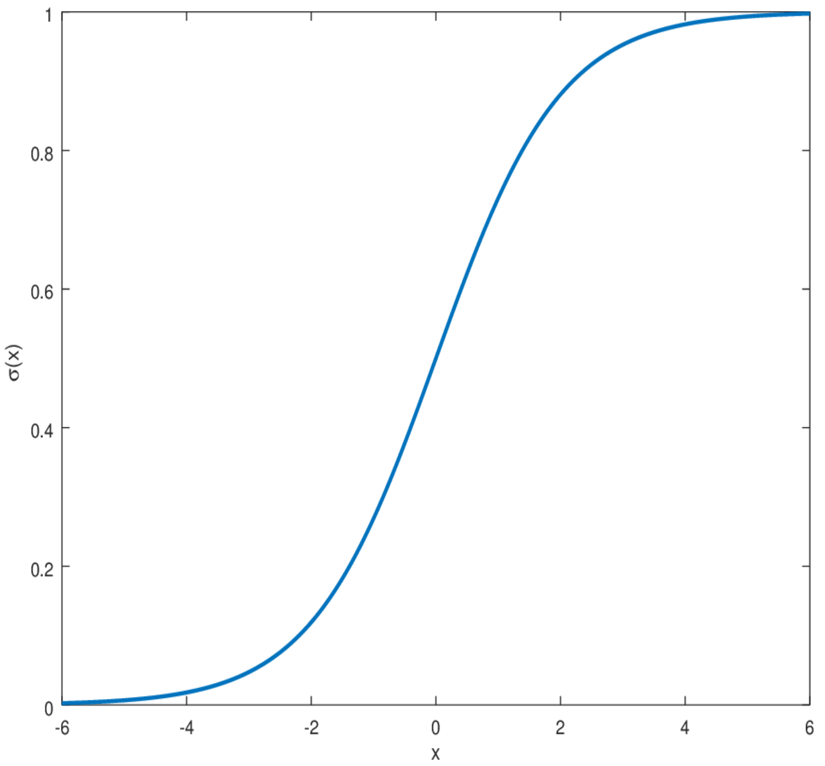
Sigmoid Function (Logistic)

- **Explanation**: The sigmoid function squashes the input values between 0 and 1, making it useful in binary classification problems where we need to produce probabilities.

- **Formula**:

$$f(x) = \frac{1}{1 + e^{-x}}$$

- **Usage**: Commonly used in the output layer for binary classification tasks, where it transforms the network's raw output into probabilities.

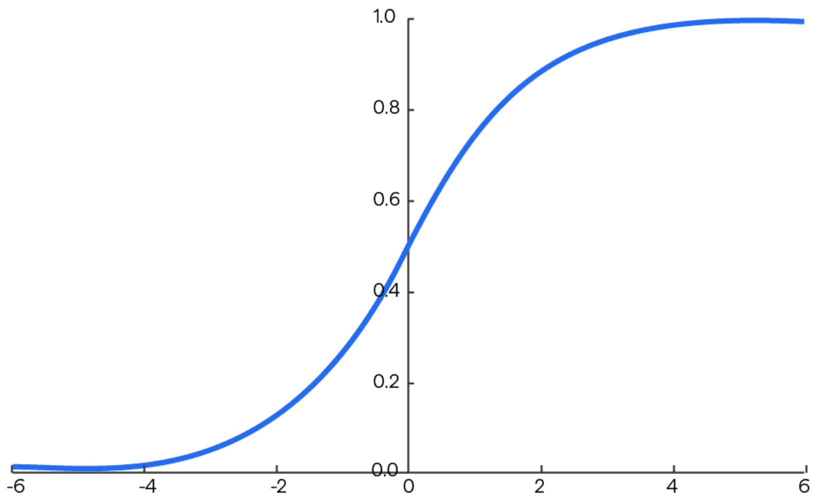


Softmax Function

- **Explanation**: The softmax function is commonly used in the output layer of neural networks for multi-class classification problems. It converts raw scores (logits) into probabilities, ensuring that the sum of the probabilities for all classes is equal to 1.

- **Formula**:

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$



- **Usage**: Softmax transforms the final layer activations into a probability distribution, allowing the model to make predictions about the likelihood of each class.

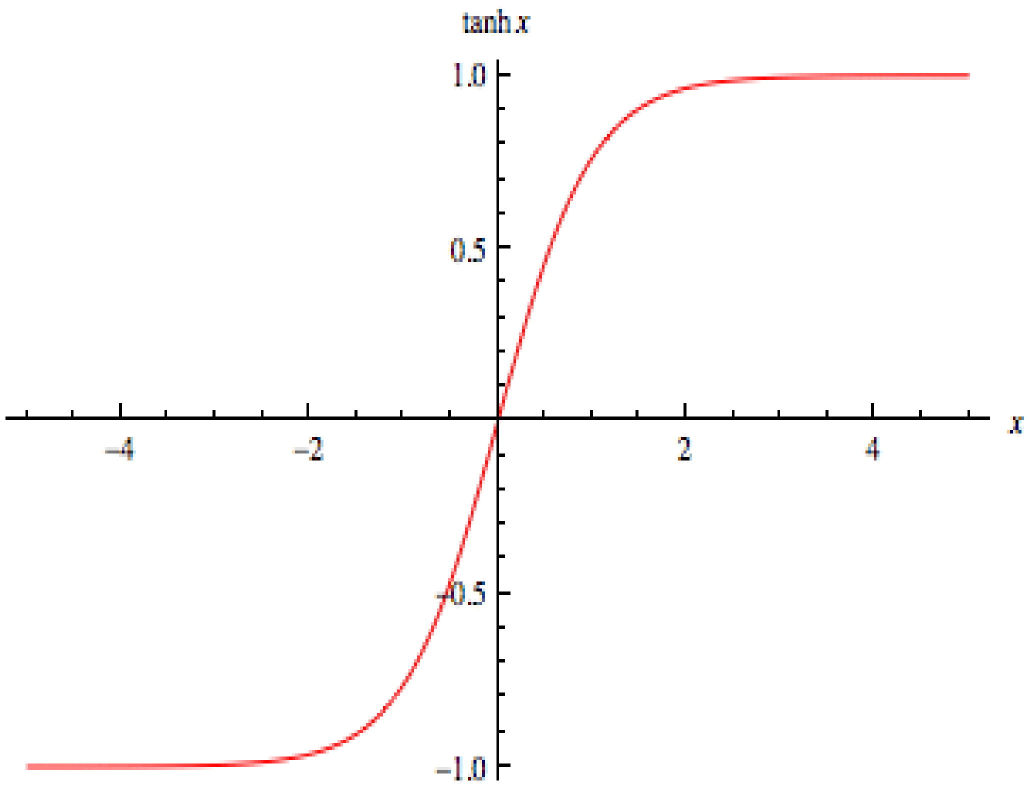
Hyperbolic Tangent Function (Tanh)

- **Explanation**: Tanh function squashes the input values between -1 and 1, which can be useful for normalization and also in classification tasks.
- **Formula**:

$$Tanh$$

$$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

- **Usage**: Similar to sigmoid, tanh is used in the hidden layers of neural networks for classification tasks, often providing better convergence properties.

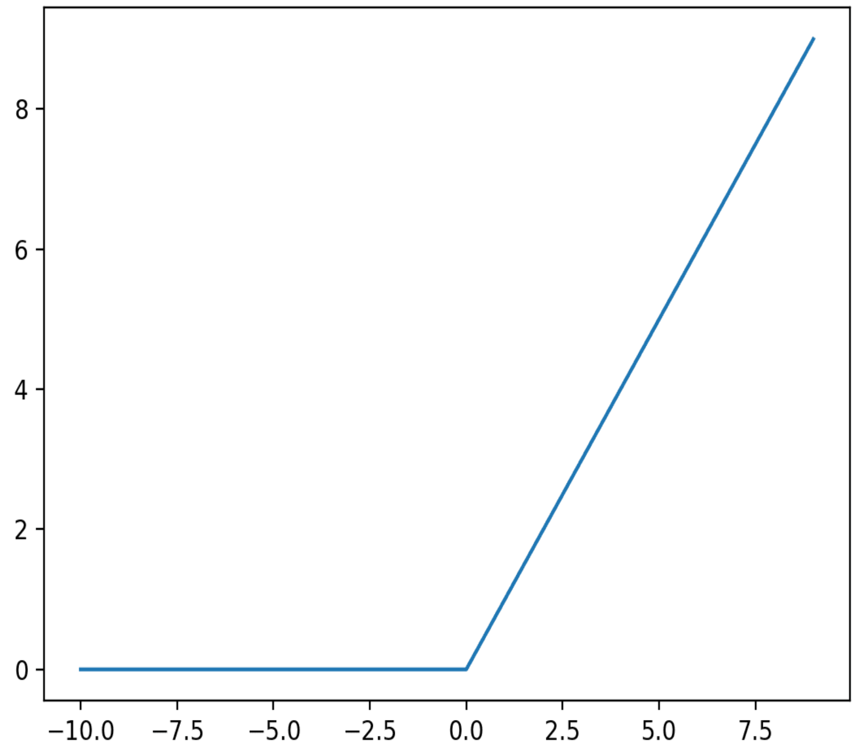


Rectified Linear Unit (ReLU)

- **Explanation**: ReLU returns 0 for negative inputs and the input value for positive inputs.
- It's widely used due to its simplicity and effectiveness.
- **Formula**:

$$f(x) = max\ (0, x)$$

- **Usage**: ReLU is widely used in hidden layers due to its simplicity and effectiveness in combating the vanishing gradient problem, common in deep networks.



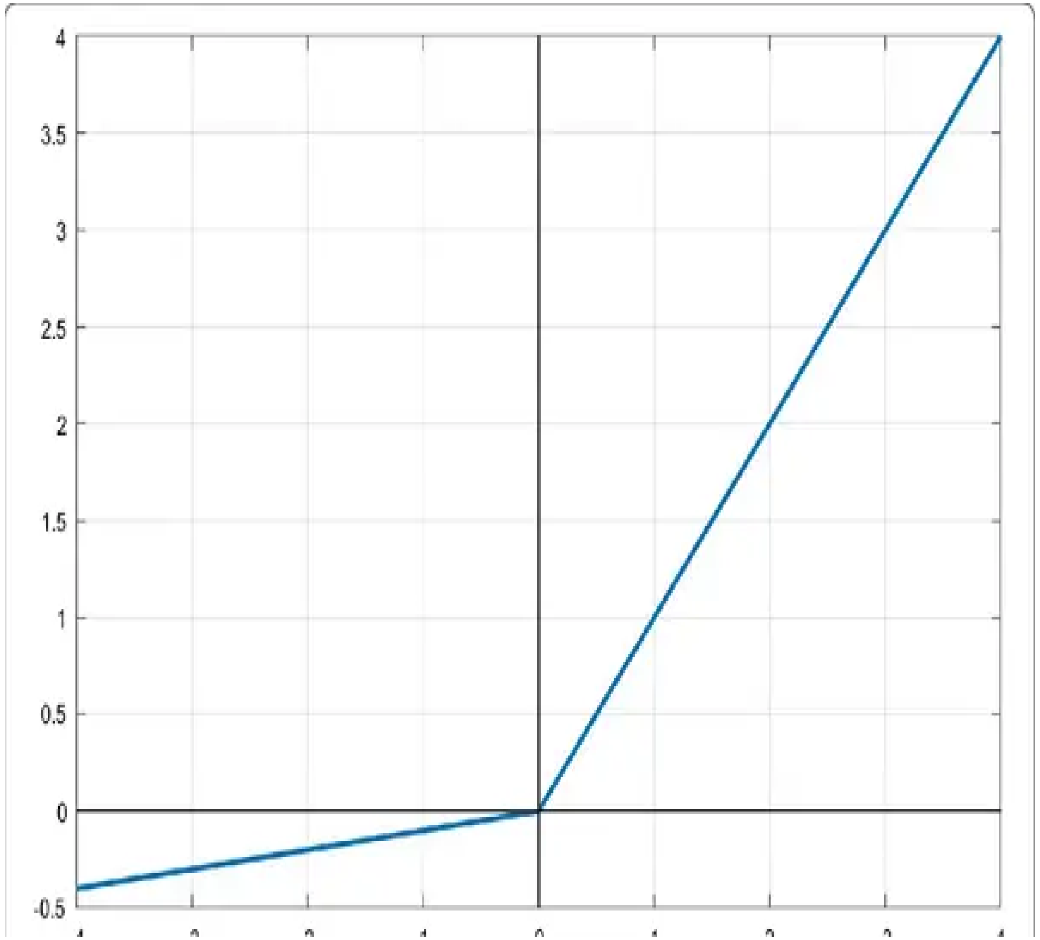
Leaky ReLU

- **Explanation**: Leaky ReLU is similar to ReLU but has a small slope for negative inputs, which helps mitigate the "dying ReLU" problem.

- **Formula**

$$f(x) = max\ (0.1x, x)$$

- **Usage**: Leaky ReLU addresses the "dying ReLU" problem by allowing a small gradient for negative inputs, which can be beneficial in training deep networks.



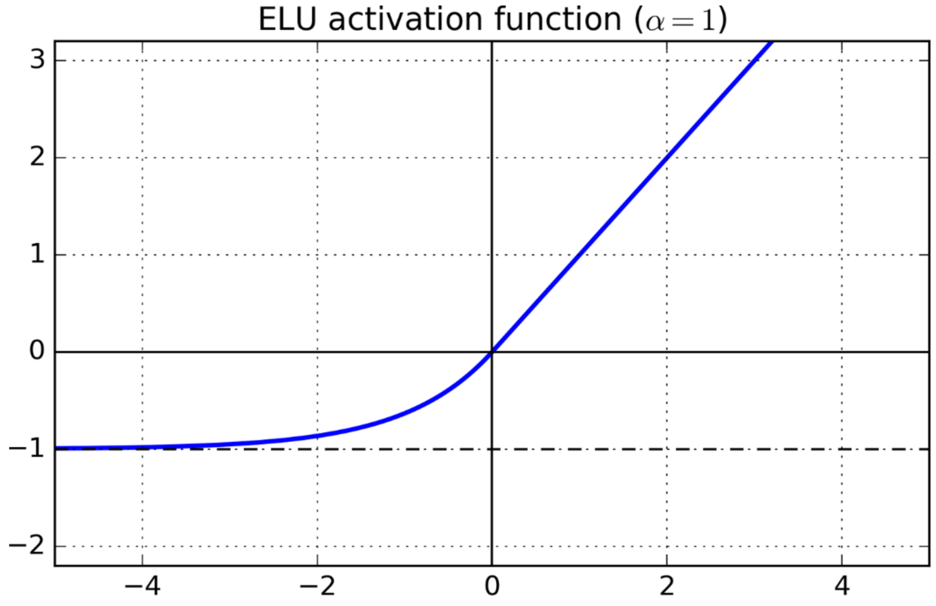
Exponential Linear Unit (ELU)

- **Explanation**: ELU is similar to ReLU for positive inputs but allows negative values with a smooth curve, aiming to address some limitations of ReLU.

- **Formula**

$$\mathrm{elu}(x) = \begin{cases} x, & x > 0 \\ \alpha\,(\exp(x) - 1), & x \leq 0 \end{cases}$$

- **Usage**: ELU mitigates the limitations of ReLU by handling negative inputs with a smooth curve, which can improve the robustness of deep networks.



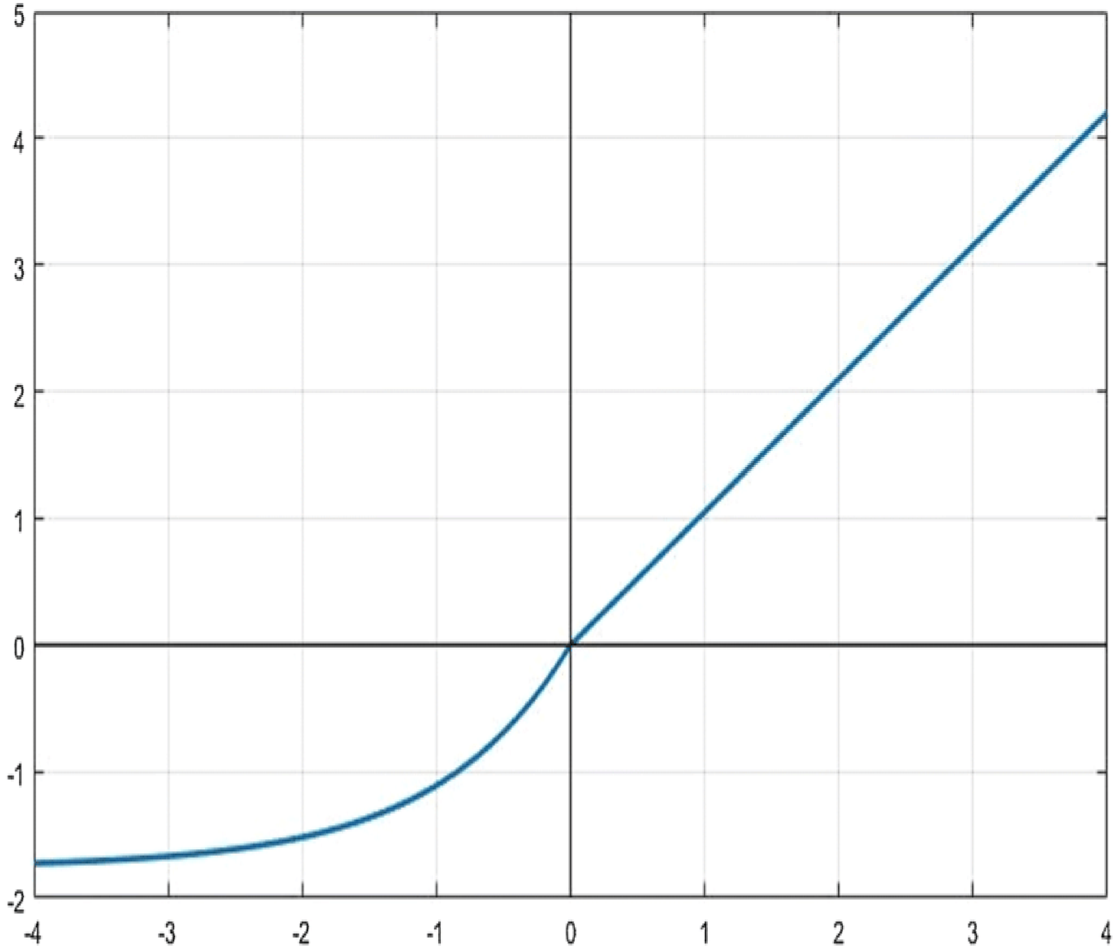
Scaled Exponential Linear Unit (SELU)

- **Explanation**: SELU is designed to maintain the mean and variance of the activations close to 0 and 1 respectively, aiding convergence.

- **Formula**

$$f(\alpha, x) = \lambda \begin{cases} \alpha\left(e^{x} - 1\right) & for\ x < 0 \\ x & for\ x \geqslant 0 \end{cases}$$

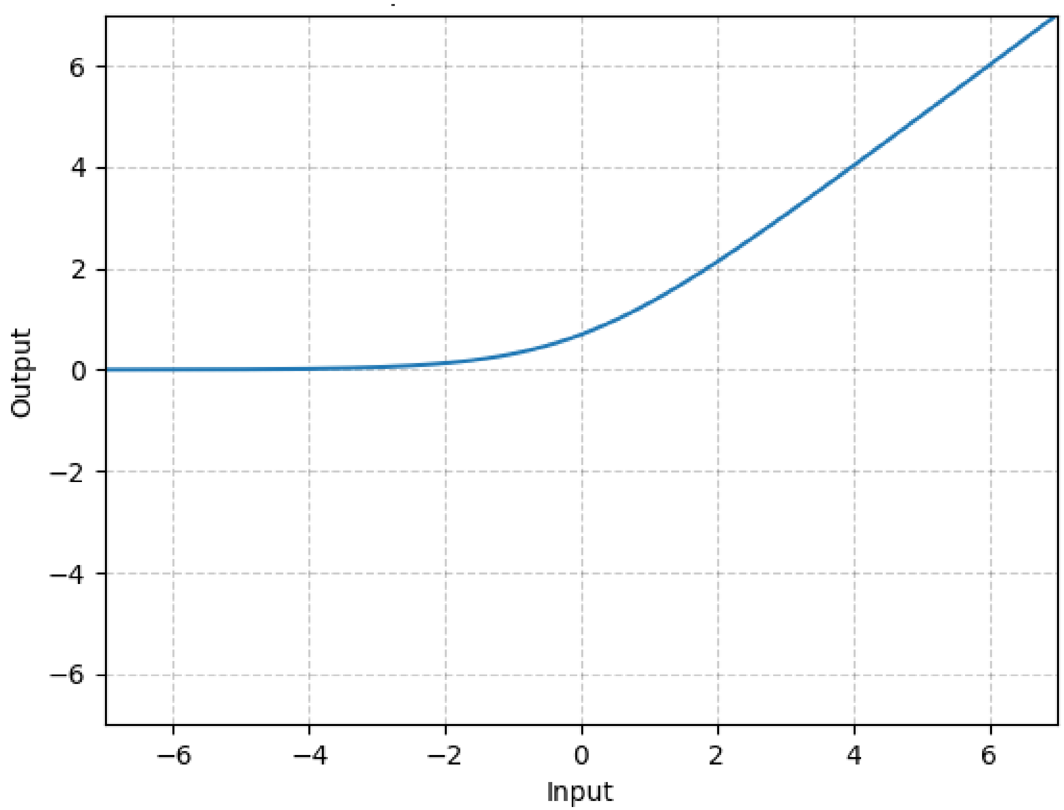
- **Usage**: SELU is designed to maintain the stability of activations throughout the network, often leading to better convergence and performance in deep architectures.



Softplus Function

- **Explanation**: Softplus is a smooth version of ReLU and can be used as an alternative activation function in some cases.
- **Formula**
- $f(x) = In(1 + e^{x})$

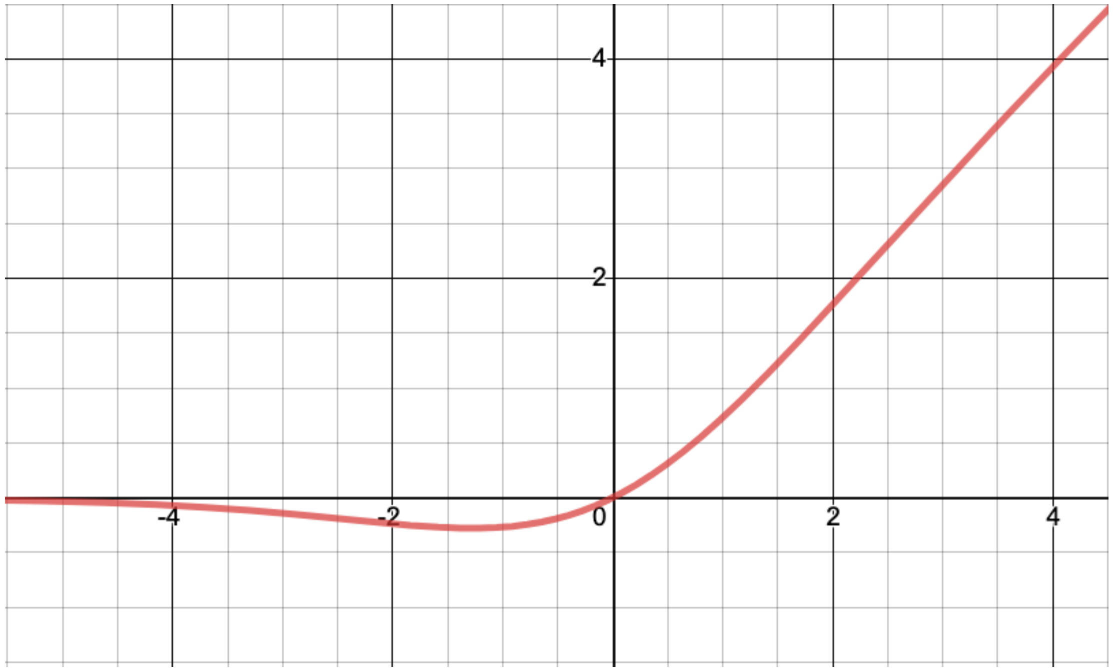
- **Usage**: Softplus is a smooth approximation of ReLU and can be used in scenarios where a differentiable activation function is required.



Swish Function

- **Explanation**: Swish function is a recently proposed activation function that tends to perform better than ReLU in certain scenarios.
- **Formula**
- $f(x) = x \cdot \text{sigmoid}(x)$

- **Usage**: Swish is an alternative to ReLU, offering potentially better performance, especially in large-scale datasets and deeper networks.



Thank You