Deep Learning for Explainable Computer-Aided Diagnosis

Presenter: Yiyang (Ian) Wang
Advisors: Dr. Daniela Raicu, Dr. Jacob Furst
May, 2020

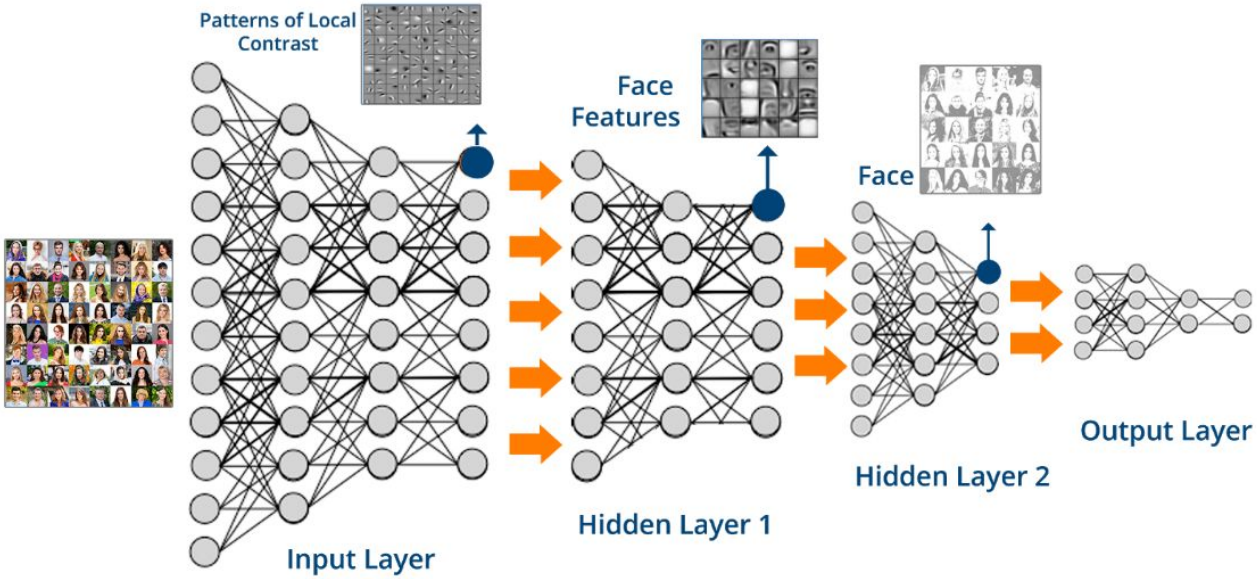
Outline

- Introduction
  - Deep Learning and Convolutional Neural Network (CNN)
- Advanced Topics
  - Transfer learning
  - One Shot Learning with Siamese Networks
  - Gradient-weighted Class Activation Mapping (Grad-CAM)
- Deep Learning Applications on Computer-Aided Diagnosis

Deep Learning and Convolutional Neural Network (CNN)

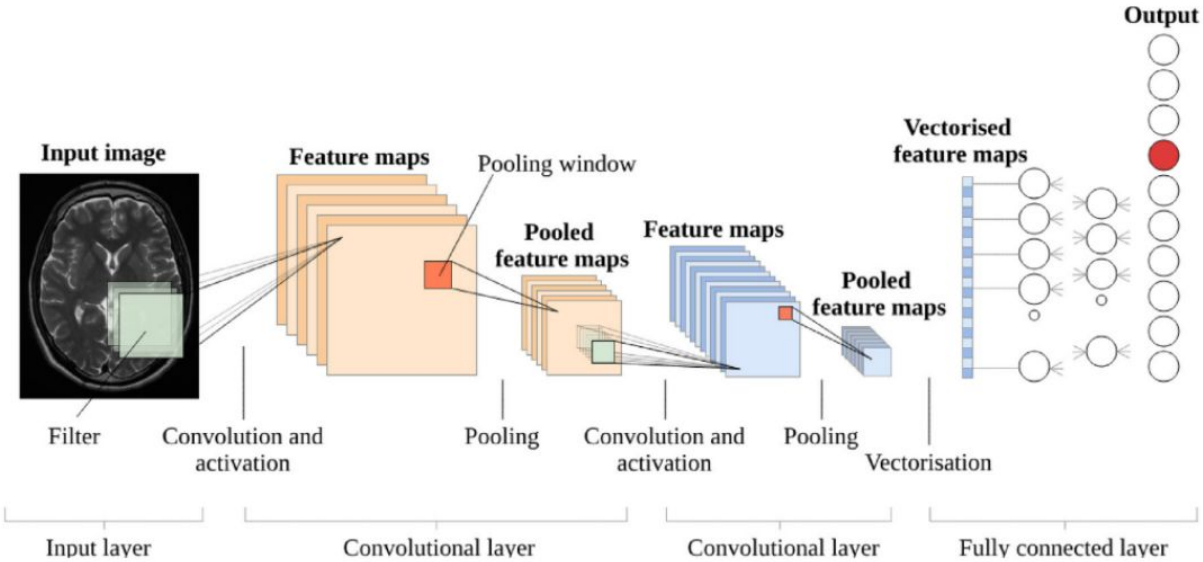
- Deep Learning is the branch of Machine Learning based on Deep Neural Networks (DNNs)
  - Neural networks with at the very least 3 or 4 layers (including the input and output layers)
- Convolutional Neural Networks (CNNs) are one of the most popular neural network architectures
  - In medical imaging the interest in deep learning is mostly triggered by CNNs

An example: Face Recognition



- At the lowest level, network fixates on patterns of local contrast as important
- The following layer is then able to use those patterns of local contrast to fixate on things that resemble eyes, noses, and mouths
- Finally, the top layer is able to apply those facial features to face templates
- A deep neural network is capable of composing more and more complex features in each of its successive layers.

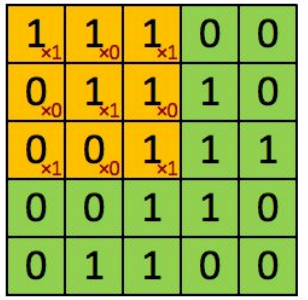
Building Blocks of CNNs



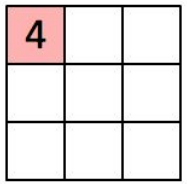
- A CNN is a particular kind of artificial neural network aimed at preserving spatial relationships in the data, with very few connections between the layers.
- The input to a CNN is arranged in a grid structure and then fed through layers that preserve these relationships, each layer operation operating on a small region of the previous layer.
- A CNN has multiple layers of **convolutions** and **activations**, often interspersed with **pooling layers**, and is trained using **backpropagation** and **gradient descent** as for standard artificial neural networks.

Building Blocks of CNN: Convolutional layers

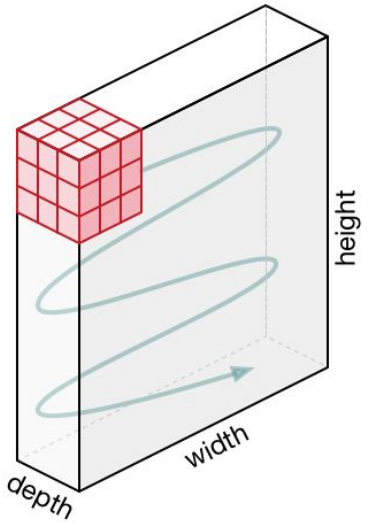
- In the convolutional layers the activations from the previous layers are convolved with a set of small **parameterized filters**, collected in a tensor $W^{(j,i)}$, where j is the filter number and i is the layer number.
- Applying all the convolutional filters at all locations of the input to a convolutional layer produces **a tensor of feature maps**.



Image

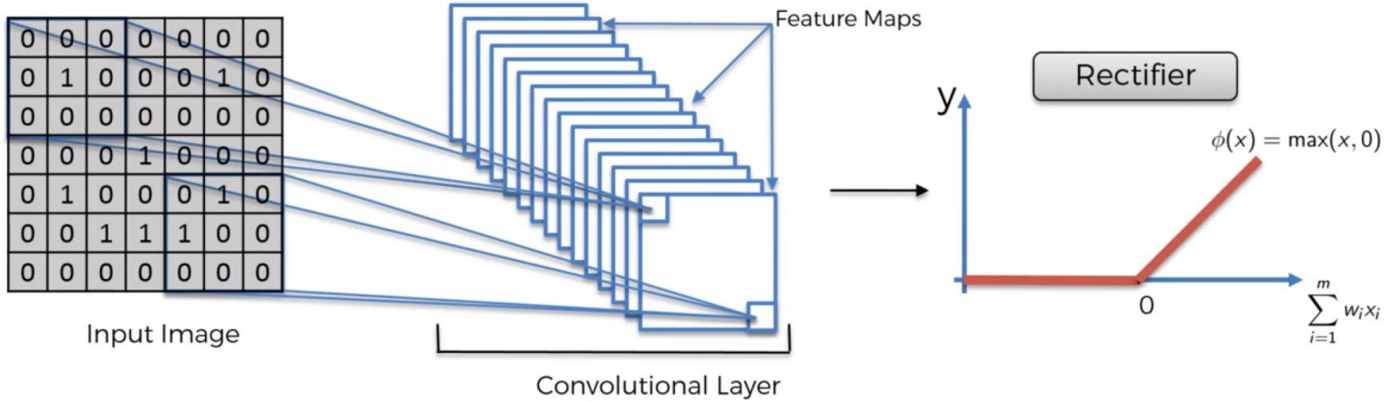


Convolved Feature



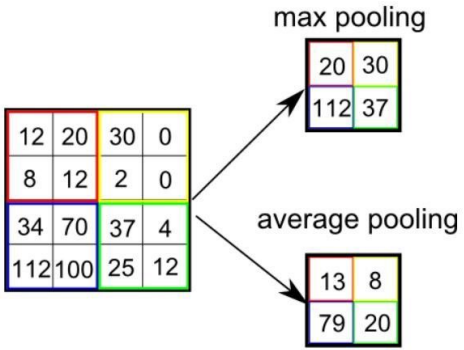
Building Blocks of CNN: Activation layer

- The feature maps from a convolutional layer are fed through **nonlinear** activation functions. This makes it possible for the entire neural network to approximate almost any nonlinear function.
- The activation functions are generally the very simple rectified linear units, or ReLUs, defined as *ReLU(z) = max(0,z)*, or its variants.



Building Blocks of CNN: Pooling Layer

- Each feature map produced by feeding the data through one or more convolutional layer is then typically pooled in a **pooling layer**.
- Pooling operations take small grid regions as input and produce single numbers for each region.
  - The number is usually computed by using the max function (max-pooling) or the average function (average pooling).
- Since a small shift of the input image results in small changes in the activation maps, the pooling layers gives the CNN some translational invariance.



Examples of New and Improved CNN Architectures

AlexNet	The network that launched the current deep learning boom by winning the 2012 Large-Scale Visual Recognition Challenge(ILSVRC) competition by a huge margin. **Notable features include the use of RELUs, dropout regularization, splitting the computations on multiple GPUs, and using data augmentation during training.** ZFNet, a relatively minor modification of AlexNet, won the 2013 ILSVRC competition.
VGG	Popularized the idea of **using smaller filter kernels and therefore deeper networks (up to 19 layers for VGG19, compared to 7 for AlexNet and ZFNet)**, and training the deeper networks using pre-training on shallower versions.

Examples of New and Improved CNN Architectures

GoogLeNet	Promoted the idea of **stacking the layers in CNNs more creatively, as networks in networks**. Inside a relatively standard architecture (called the stem), GoogLeNet contains multiple inception modules, in which multiple different filter sizes are applied to the input and their results concatenated. This multi-scale processing allows the module to extract features at different levels of detail simultaneously. GoogLeNet also popularized the idea of not using fully-connected layers at the end, but rather global average pooling, significantly reducing the number of model parameters. It won the 2014 ILSVRC competition
ResNet	Introduced **skip connections, which makes it possible to train much deeper networks**. A 152 layer deep ResNet won the 2015 ILSVRC competition, and the authors also successfully trained a version with 1001 layers. Having skip connections in addition to the standard pathway gives the network the option to simply copy the activations from layer to layer (more precisely, from ResNet block to ResNet block), preserving information as data goes through the layers. Some features are best constructed in shallow networks, while others require more depth. The skip connections facilitate both at the same time, increasing the network's flexibility when fed input data. As the skip connections make the network learn residuals, ResNets perform a kind of boosting.

Transfer Learning

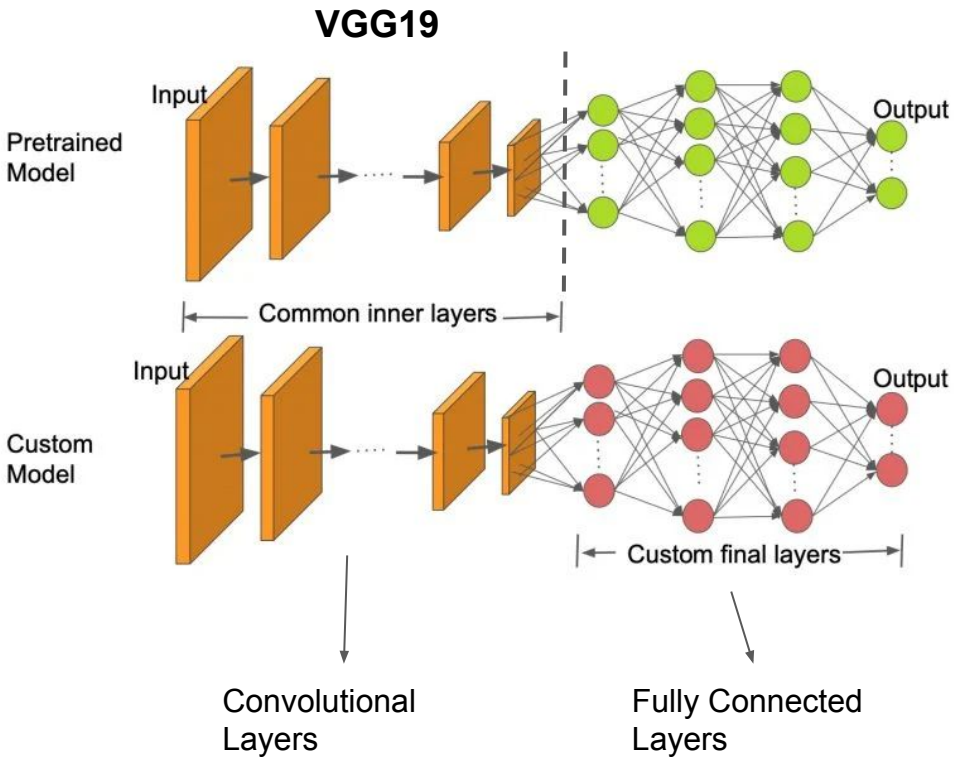
Transfer Learning

- Transfer learning is a popular method in computer vision because it allows us to build accurate models in a timesaving way.

- With transfer learning, instead of starting the learning process from scratch, we start from patterns that have been learned when solving a different problem.
  - This way we leverage previous learnings and avoid starting from scratch.

- In computer vision, transfer learning is usually expressed through the **use of pre-trained models.** A pre-trained model is a model that was trained on a large benchmark dataset.

Transfer Learning



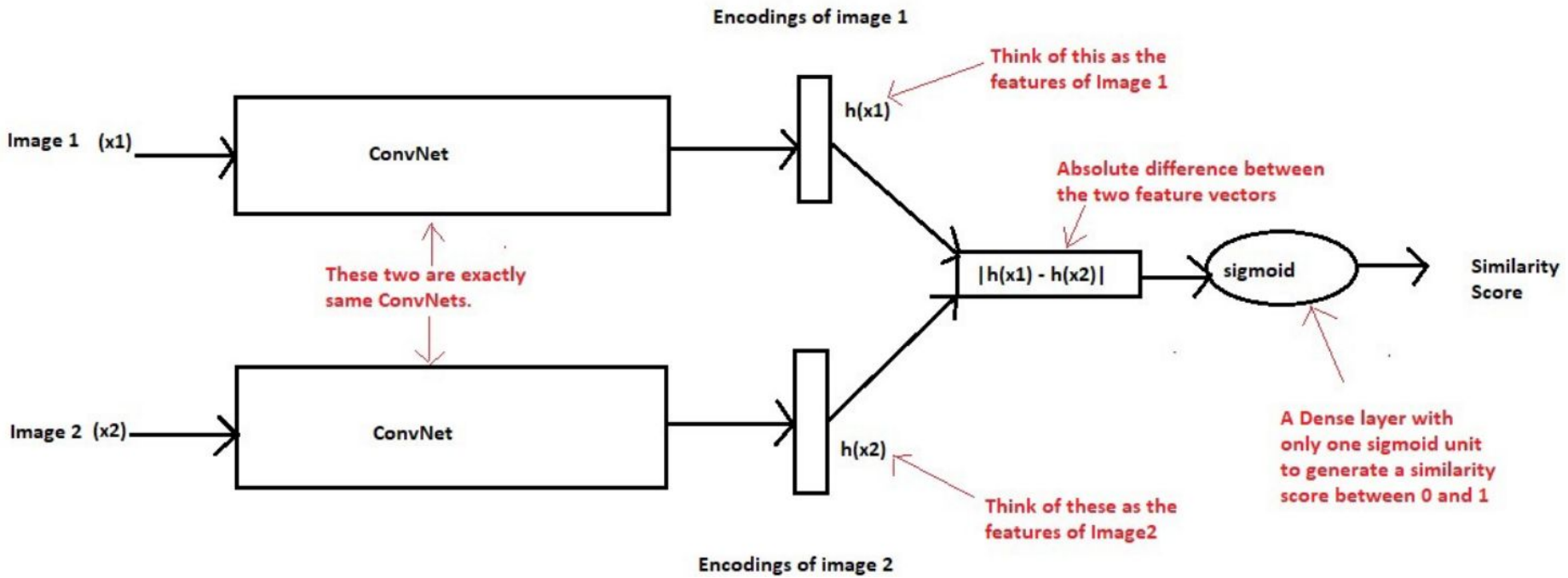
- Pretrained Model: **VGG-19** on **ImageNet** [1]
- ImageNet: 1.2 million images with 1000 categories (dog, cat, car *et al.*)
- The output of original pretrained model (green) is the probability of each 1000 categories
- The output of custom model (red) is the probability of each N categories in our own data set
- **During the training, we train the last few convolutional layers and the fully connected layers**

One shot Learning

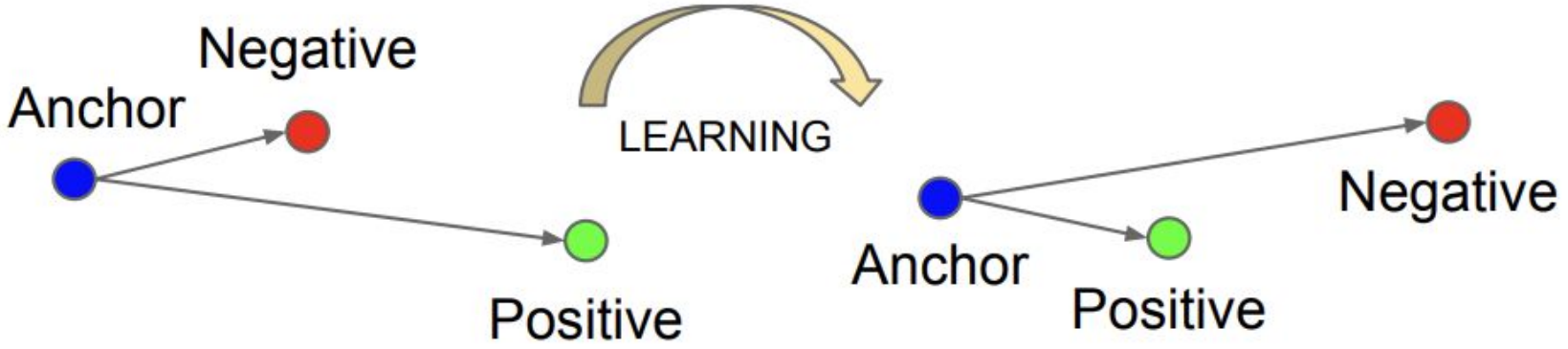
One Shot Learning with Siamese Networks

- In a one shot classification: we require only **one training example** for **each class.**
- **A Siamese network** is an architecture with **two parallel neural networks**, each taking a different input, and whose outputs are combined to provide some prediction. [2]
- The deep CNNs are first trained to discriminate between examples of each class. The idea is to have the models learn feature vectors that are effective at extracting abstract features from the input images.
- The models are then re-purposed for verification to predict whether new examples match a template for each class.
    - Specifically, **each network produces a feature vector for an input image**, which are then compared using the **L1 distance and a sigmoid activation**.

Siamese Networks



Siamese Networks: Triplet Loss



- The loss function penalizes the model such that the distance between the matching examples is reduced and the distance between the non-matching examples is increased.

Gradient-weighted Class Activation Mapping (Grad-CAM)

Visualization via Gradient-weighted Class Activation Mapping (Grad-CAM) [3]

- Grad-CAM uses the gradient information flowing into the last convolutional layer of the CNN to understand each neuron for a decision of interest.
- To obtain the class discriminative localization map for any class c, we first compute the gradient of the score for the class c, y^c (before the softmax) with respect to feature maps A^k of a convolutional layer.
- These gradients flowing back are global average-pooled to obtain the neuron importance weights ak for the target class.

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

Visualization via Gradient-weighted Class Activation Mapping (Grad-CAM)

- After calculating α for the target class c, we perform a weighted combination of activation maps and follow it by ReLU

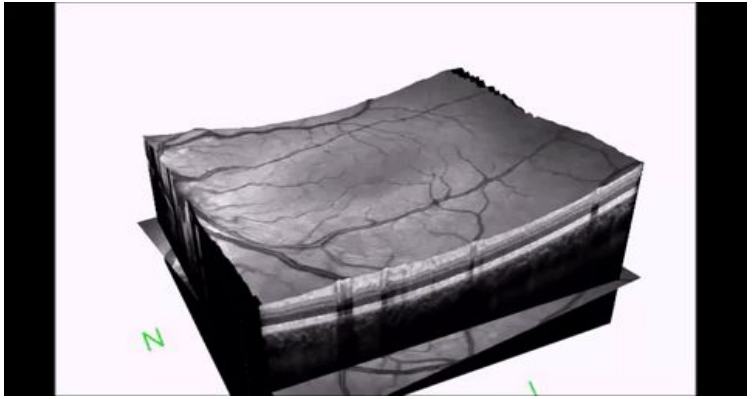
$$L^{c}_{\text{Grad-CAM}} = ReLU \underbrace{\left(\sum_{k} \alpha^{c}_{k} A^{k} \right)}_{\text{linear combination}}$$

- This results in a coarse heatmap of the same size as that of the convolutional feature maps.
- We apply ReLU to the linear combination because we are only interested in the features that have a positive influence on the class of interest.

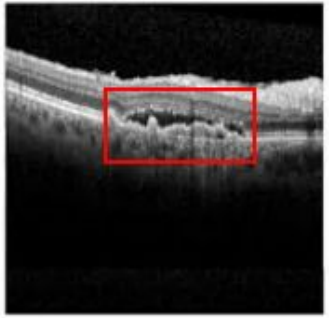
Deep Learning Applications on Computer-Aided Diagnosis

Application 1: Classifying OCT Images with Age-related Macular Degeneration (AMD) Biomarkers

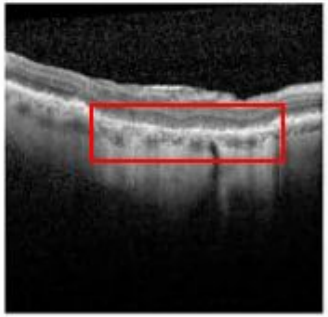
- AMD is a major health burden that can lead to irreversible vision loss in the elderly population.
- Our task is to classify images with three types of AMD biomarkers (Wet, Dry, Drusen) and healthy images



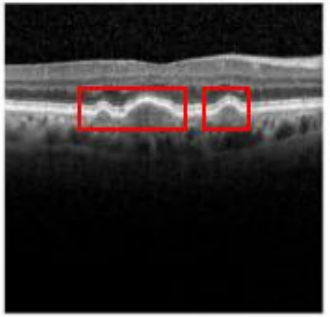
- Horizontal scans of the eye (B-scans)
- About 25 are kept for analysis



Wet



Dry



Drusen

Application 1: VGG-19 Architecture-Convolutional Layers

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 256, 256, 3)	0
block1_conv1 (Conv2D)	(None, 256, 256, 64)	1792
block1_conv2 (Conv2D)	(None, 256, 256, 64)	36928
block1_pool (MaxPooling2D)	(None, 128, 128, 64)	0
block2_conv1 (Conv2D)	(None, 128, 128, 128)	73856
block2_conv2 (Conv2D)	(None, 128, 128, 128)	147584
block2_pool (MaxPooling2D)	(None, 64, 64, 128)	0
block3_conv1 (Conv2D)	(None, 64, 64, 256)	295168
block3_conv2 (Conv2D)	(None, 64, 64, 256)	590080
block3_conv3 (Conv2D)	(None, 64, 64, 256)	590080
block3_conv4 (Conv2D)	(None, 64, 64, 256)	590080
block3_pool (MaxPooling2D)	(None, 32, 32, 256)	0
block4_conv1 (Conv2D)	(None, 32, 32, 512)	1180160
block4_conv2 (Conv2D)	(None, 32, 32, 512)	2359808
block4_conv3 (Conv2D)	(None, 32, 32, 512)	2359808
block4_conv4 (Conv2D)	(None, 32, 32, 512)	2359808
block4_pool (MaxPooling2D)	(None, 16, 16, 512)	0
block5_conv1 (Conv2D)	(None, 16, 16, 512)	2359808
block5_conv2 (Conv2D)	(None, 16, 16, 512)	2359808
block5_conv3 (Conv2D)	(None, 16, 16, 512)	2359808
block5_conv4 (Conv2D)	(None, 16, 16, 512)	2359808
block5_pool (MaxPooling2D)	(None, 8, 8, 512)	0

Total params: 20,024,384

Freeze during training process

Unfreeze during training process

Application 1: VGG-19 Architecture-Fully Connected Layers

block5_pool (MaxPooling2D)	(None, 8, 8, 512)	0
flatten_1 (Flatten)	(None, 32768)	0
dense_1 (Dense)	(None, 1024)	33555456
dense_2 (Dense)	(None, 1024)	1049600
dense_3 (Dense)	(None, 1024)	1049600
dense_4 (Dense)	(None, 512)	524800
dense_5 (Dense)	(None, 4)	2052

Total params: 56,205,892
Trainable params: 45,620,740
Non-trainable params: 10,585,152

Add our own dense layers

Produce **probability of 4 categories** in our dataset

Application 1: Confidence and Reliability

- Given k class labels (k=4 in our case), we define Confidence and Reliability as:

  Confidence = max (P_1, P_2, P_3… P_k)

  Reliability = Confidence - max (P_1, P_2, P_3… P_{k-1})

  (where P_i is the prediction probability of class i; the notation for reliability above assumes that P_k is the largest probability across all class k probabilities)

Application 1: Data Distribution

	Training	Testing	# Patients
Drusen (Under-sampling)	105	27	6
Dry	99	25	13
Mixed_Healthy (Under-sampling)	97	25	6 Healthy patients 14 unhealthy patients
Wet	94	25	11

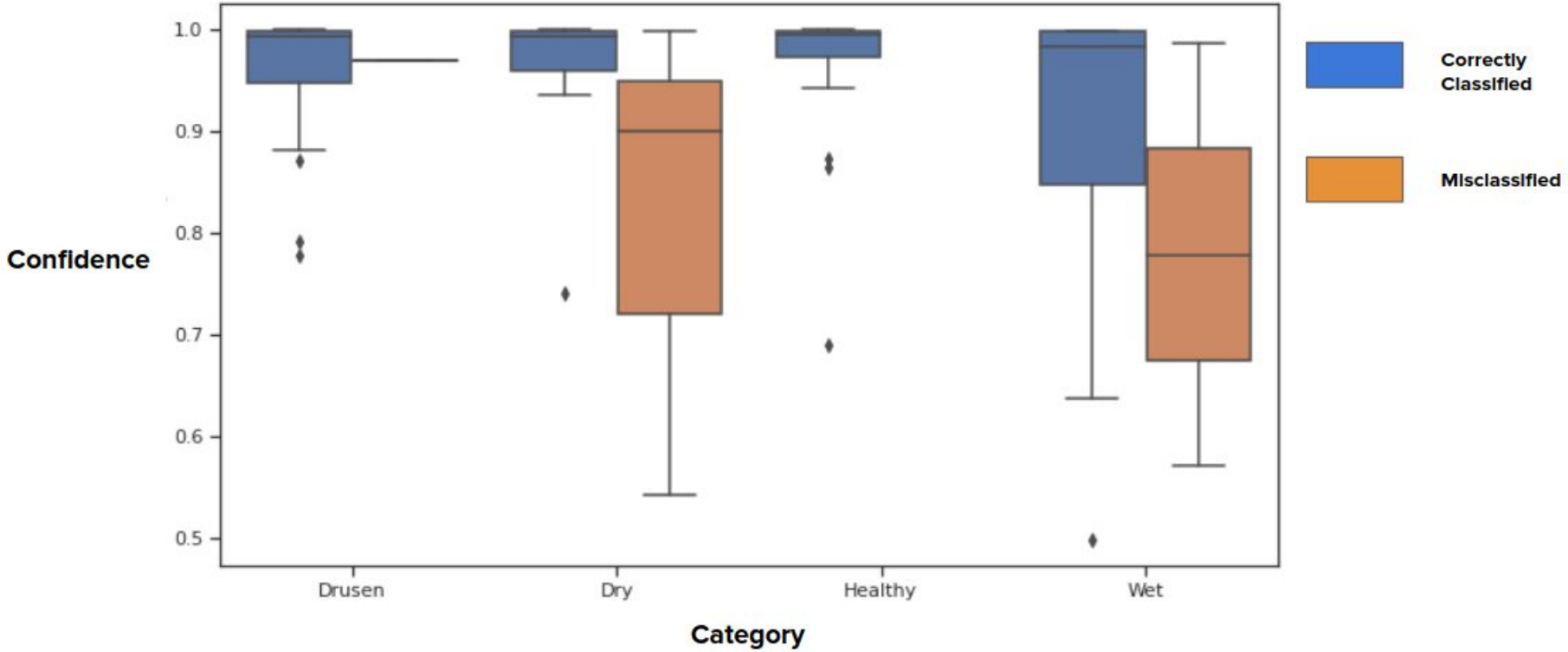
We have very limited Dry and Wet images; therefore, we implemented down-sampling for Drusen and Healthy classes. All the Dry and Wet images come from the “last visit”.

Application 1: Classification Results on Testing Data set

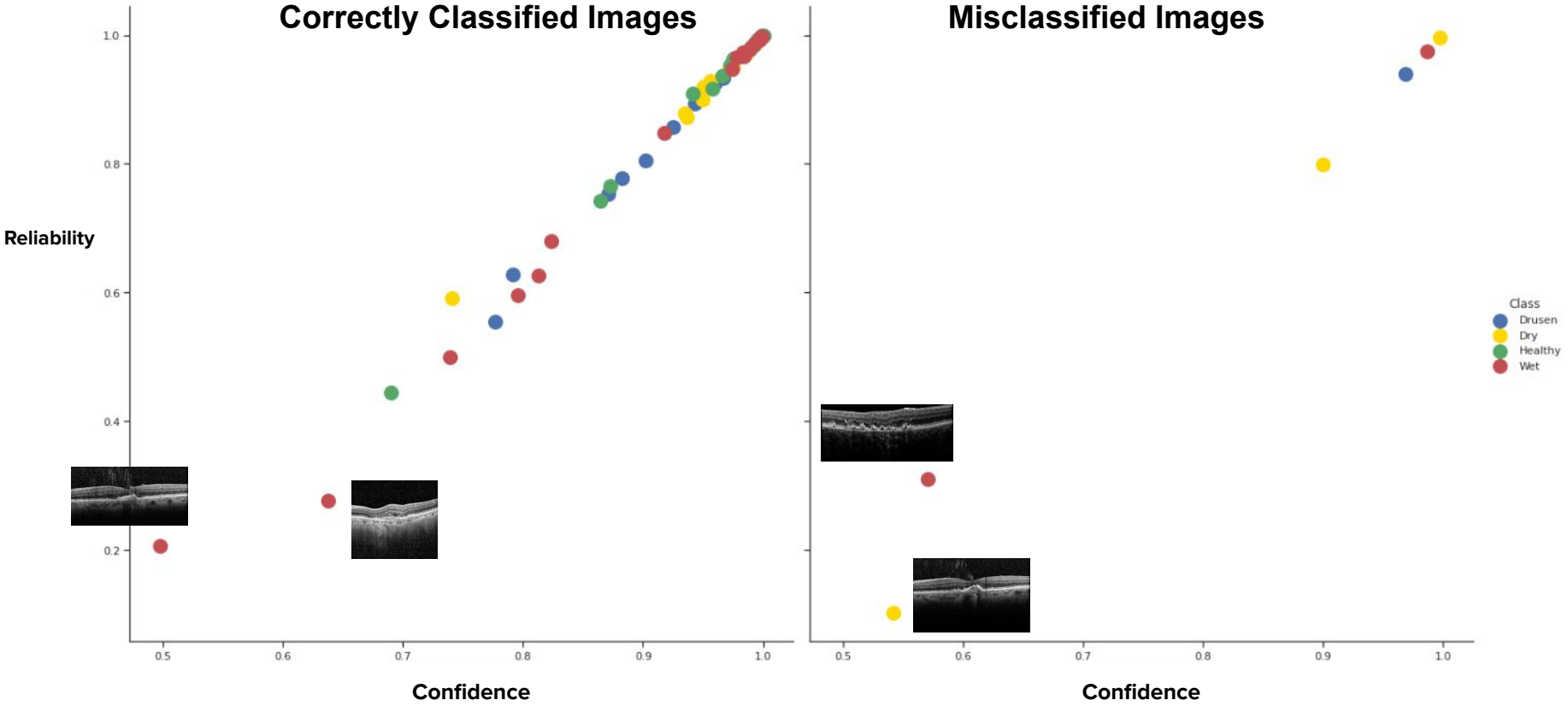
		Drusen (Under-sampling)	Dry	Mixed-Healthy (Under-sampling)	Wet
Actual	Drusen (Under-sampling)	**26**	0	1	0
	Dry	2	**22**	0	1
	Mixed-Healthy (Under-sampling)	0	0	**25**	0
	Wet	0	2	0	**22**

Overall Accuracy: 93.14%

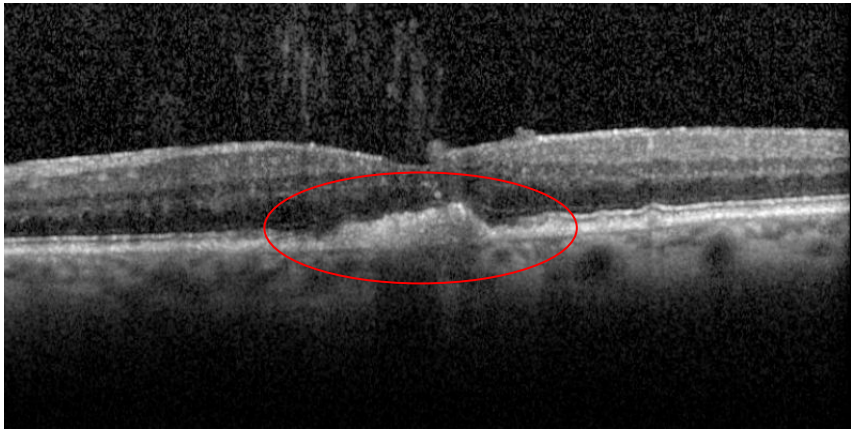
Application 1: Confidence

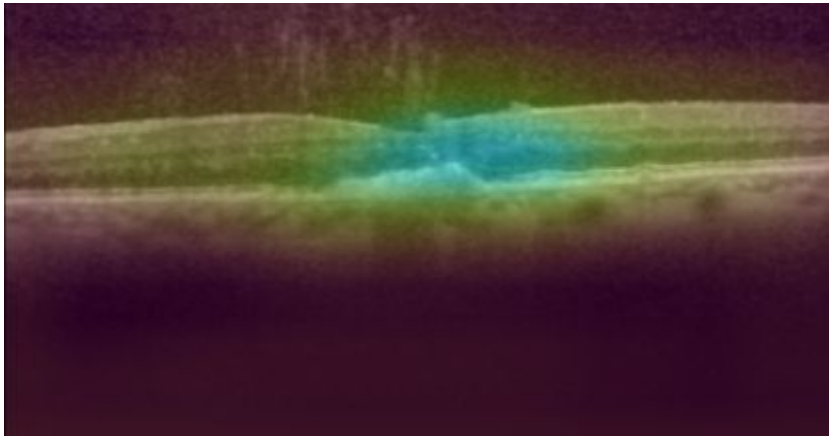


Application 1: Confidence and Reliability



Application 1: A Correctly Classified Wet Image

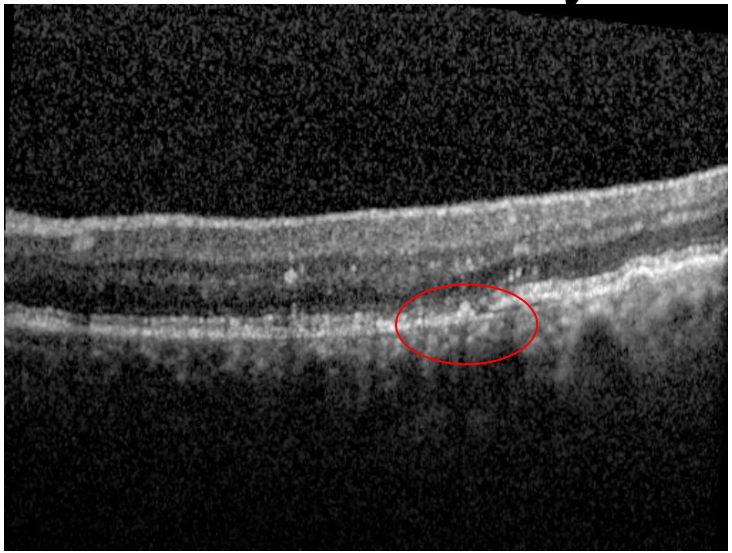




- Red circle represents the Wet biomarker picked by a human annotator
- Deep Learning Prediction:
  49.75% Wet;
  29.08% Dry

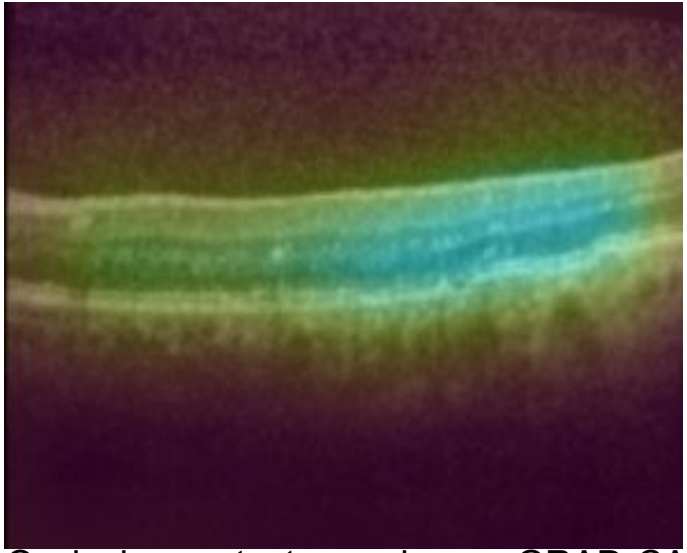
- Occlusion test using GRAD-CAM algorithm. The highlighted yellowish area represents the most important region detected by the deep learning algorithm to predict this image as “Wet”.
- The deep learning algorithm picked the same region with human annotators

A Misclassified Dry Image



- Red circle represents indicates the Dry biomarker picked by a human annotator
- Deep Learning Prediction:
  89.96% Drusen;
  9.97% Dry

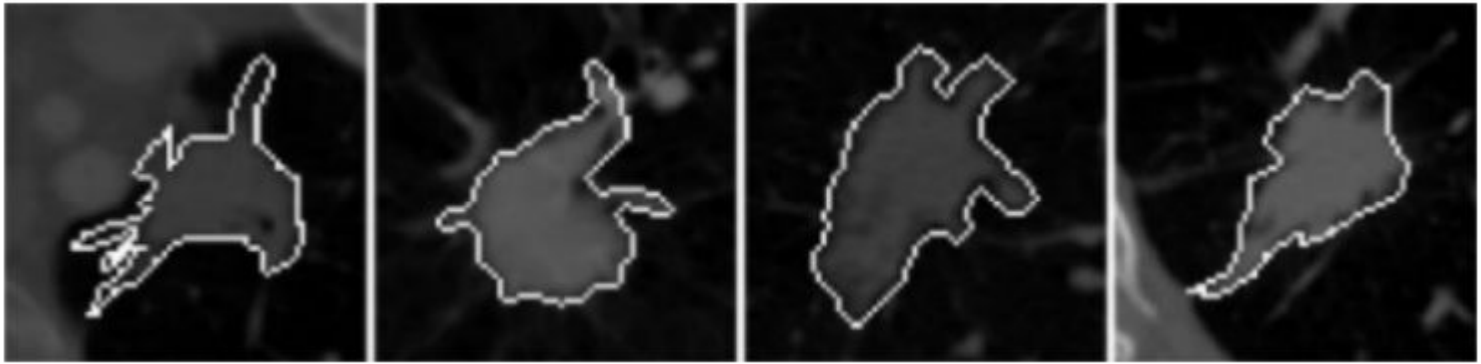
Actually this is a correct prediction



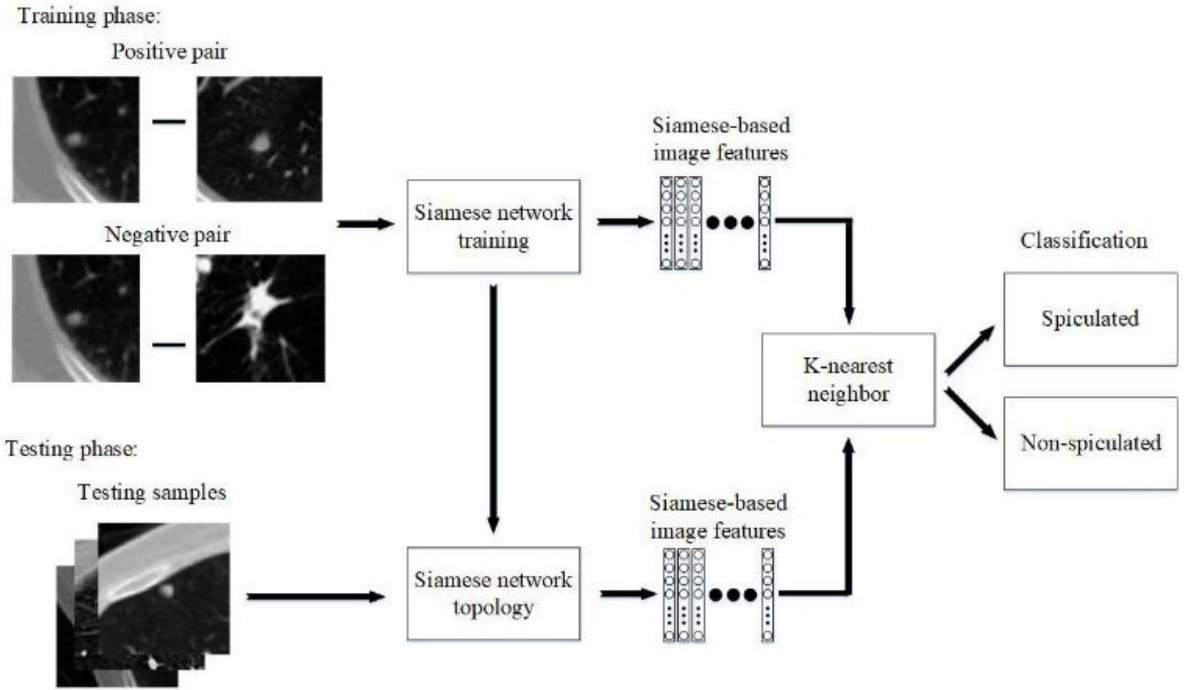
- Occlusion test using GRAD-CAM algorithm. The highlighted yellowish area represents the most important region detected by the deep learning algorithm to predict this image as “Drusen”.
- The deep learning algorithm actually picked the **correct region for drusen, but did not find the correct dry biomarker**

Application 2: Lung Nodule Classification

- The NIH/NCI Lung Image Database Consortium (Armato et al, 2004)
  - 1010 patients
  - 2680 distinct nodules
  - Cropped nodule size: 71 by 71
  - Target label: Spiculation (binary)



Application 3: Siamese Convolutional Neural Network



- 2 convolutional layers, 1 dense layer
- 30 kernels in each convolutional layer
- 128 neurons in the dense layer
- ReLU activation function following each layer
- Loss function: Triplet loss
- **Training Data**
  - 400 nodules
  - 39,800 positive pairs
  - 40,000 negative pairs
- **Testing Data**
  - 80 nodules
  - 1,560 positive pairs
  - 1,600 negative pairs

Application 3: Classification Result

K = 9 Method	Accuracy (%)	Class	Sensitivity (%)	Specificity (%)
Siamese-based feature (training)	89.23 ± 0.94	1	87.2 ± 1.3	91.3 ± 1.3
		2	91.3 ± 1.3	87.2 ± 1.3
Siamese-based feature (test)	**84.18 ± 1.14**	**1**	**84.8 ± 1.5**	**83.7 ± 2.0**
		2	**83.7 ± 2.0**	**84.8 ± 1.5**
Designed features (test)	78.44 ± 1.55	1	83.3 ± 2.1	73.5 ± 2.4
		2	73.5 ± 2.4	83.3 ± 2.1

95% confidence intervals over the 20 experiments for accuracy and sensitivity for the best k-NN results (k = 9) from the range of k=1 to k=11.

Thank you!

To know more information about our lab, please Google *'Visual Computing DePaul'*

http://facweb.cs.depaul.edu/research/vc/

If you have any questions regarding this presentation, please email to:

ywang192@depaul.edu

References

[1] Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In 2009 IEEE conference on computer vision and pattern recognition, pp. 248-255. Ieee, 2009.

[2] Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." In ICML deep learning workshop, vol. 2. 2015.

[3] Selvaraju, Ramprasaath R., Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. "Grad-cam: Visual explanations from deep networks via gradient-based localization." In Proceedings of the IEEE international conference on computer vision, pp. 618-626. 2017.