**Today’s Agenda:**

Introduction content:

This seminar will be particularly valuable for anyone interested in using MATLAB to process, visualize, and quantify biomedical imagery.  Rather than focus on extracting information from a few homogeneous images, we will introduce a typical real-world challenge, and discuss approaches to managing and exploring collections of widely heterogeneous images.  We will describe user interfaces that simplify the exploration and algorithm development processes, and demonstrate their utility in identifying and quantifying scientific or clinically relevant insights.

We will then focus on the extraction of features from images, and the use of machine learning algorithms to classify images based on their content.

The agenda today is to go over two real-world examples in MATLAB. The Demos are Scene Classification and Object Classification. In other words, where am I? and what am I looking at?

So Let’s jump right in. Our first example today will be to automatically classify what scene we are in. And for this demo we’re going to be using a Machine learning approach. So in order to get calibrated, let’s define what a machine learning workflow looks lie, and later we’ll define a deep learning workflow and the differences between the two.

Machine Learning Workflow. Let’s say I want to classify 3 different categories of objects. And for this example I chose three different categories of animals. I first want to take a training image and extract features from it. And features can be anything from corners, edges, color, anything that can represent the object that is invariant to variations in lighting conditions and can also handle changes to images like rotation and scale.

Once we extract those features, we want to add them to our machine learning model. And we want to have many examples of the object we’re looking to recognize. So if I’m looking to classify dogs, cats and turkeys, we want to extract all of those features and add it to our model.

We know what class these features represent, so we want to train our model to fit these features into 3 distinct buckets. We now have a trained model, so when a test image comes in, we want to extract those same features, and predict what category those features best represent. So keep this model in mind, let’s jump into MATLAB to perform this feature extraction and classification.

**Demo #1:**

The goal of this demo is to Identify what scene an image is from with the goal of answering the question: “Where am I?”

These locations are fairly diverse and have lots of different variations, so we want a lot of sample from each category to help correctly differentiate between them.

**Image Datastore:**

I want to bring up the challenge of how we are supposed to read in all of these images. We don't want to have to loop through each file in a directory and check that it's a proper image format. Instead, we can use imageDatastore to automatically read in all the images in a directory.

We can specify whether we want to read subfolders or not. And we can label our data based on the folder names So if I already have my data in subfolders with the proper names, imageDatastore will read in my data from these folders and label them accordingly

Note another benefit of using imageDatastore is that it can handle massive amounts of data, and handle reading in the images even if they all don't fit into memory at once, which makes it great for machine learning and deep learning tasks [ and I'll be using it throughout all of the demos today. ]

Once I've specified the location of my data, I can count the number of images in each directory.

**“Features”**

Now I want to take these images in my training set and extract features from them.

I have lots of options for extracting features, I can use edges or corners, SURF Features, the list goes on, but I want to highlight one feature extractor call bagOfFeatures.

Bag of Features, also could be defined as bag of visual words. Let's take our beach for example,

If I asked you to describe what you might see at the beach, what words would you use? Maybe the water, or sand, or a beach chair... How about a restaurant? Food table chairs.

These are my visual words. How likely is it that I see these words for these scenes? Maybe the probabilities look something like this for a beach and this for a restaurant. So we count the number of times we see these words. And based on that frequency we can describe different scenes. We can look at our code and have two lines of code that does this for us. BagofFeatures() which describes the most important words, and Encode() which counts the number of times we see these words.

In the time it took me to say all of that, the creation of the bag of words is done. Now we can visualize those features. I’ve taken a random image from each of the 4 categories, and we’ve asked for 250 words to describe these scenes. On the right you can see the occurrences of those words.

Now we want to define a classifier that takes those occurrences and is able to differentiate the differences between the 4 classes.

There are a lot of different machine learning techniques we can use to create a classifier, and to help us determine what works for our data interactively, we can use an APP call the Classification Learner App

**Classification Learner App**

There are a lot of different options in this app, so I encourage you to try this out on your own data since we don't have time to go through all of the options in detail, but what I want to point your attention to is the list of all the classifiers.

The one question I get a lot is 'how do I know which classifier to use?' or 'What will work best on my data?'

The answer is: I don't know. Everyone's data is different so I can't say: Always use a quadratic SVM, if that worked every time, we would need so many options...

Instead, I can tell you to try a bunch of diff classifiers and see what works best for your data. So we can try a quadratic SVM and see how that performs. We can also try all of the SVMs, and all of the KNNs as well. So at the end I want to pick the classifier with the highest accuracy, which happens to be the medium Gaussian SVM.

One other feature in this app is the Confusion matrix. If the classifier was working perfectly, 100% accurate, we would see all green along the diagonal. Which would mean that every time it predicted a scene, it got it right. Instead, we see red on the outside of the diagonal. So for example, there were 26 cases where it predicted a beach, but it was actually a field. And this tells us that sometimes it sometimes mistakes beaches for fields, and this gives us an indication of what features we might want to use or avoid, and what we should be on the lookout for in our data.

I’m going to export that model, but do keep in mind that we can also general MATLAB code from the app. So if I’m looking to repeat this process over and over, I can automate this process and not have to use the interaction of the app every time.

Once I’ve exported the model from the app, it’s given me the way to predict on new data, and I can use that line of code to predict on my test data. So in this case, I’m extracting the same features from my test set, and I’m going to predict using the trained classifier.

So for each of my images in my test set, I get an accuracy of around 70%. Keep this number in mind because we’re going to revisit this example later and see if we can increase that number.

I can test my classifier visually as well, so I can pick a random image from my test set and I can see visually which images the classifier is getting right and wrong Maybe based on that information, I can decide to use different features or a different classifier to potentially increase the accuracy.

**End of Demo #1:**

Let me give you 3 things to take away from that demo.

1. The first is the Classification Learner App. You don’t need to be a machine learning expert to use this app. You don’t need to know which classifier is going to work best. You can try many or all of them and see which one works best for your data
2. BagOfFeatures was the feature extraction method we used. You can call this in one line of MATLAB code
3. Try ImageDatastore to be able to read in directories of data without worrying about it all fiting into memory.

**Moving on.**

You’ve already seen this slide.

How does this workflow (Machine Learning) compare to a deep learning one?

We’re going to take everything related to feature extraction and classification and bundle it into a CNN (convolutional neural network). This is one of the most popular deep learning architectures.

This will perform end-to-end feature learning and classification for us. So we will feed our images directly into the network and it will learn the best features and perform classification.

(Slide 30) So I want to discuss two approaches to deep learning. The first is to create your own neural network from scratch. And the second is to take a pre-trained model and use that to perform a new classification task. So in the second example, I have a network that someone else has already trained to classify many objects. And now I want to refine the network to perform a new task.

(Slide 31) Training a network from scratch is one approach to deep learning. You would be responsible for setting up all of the layers of the network and the weights, and MATLAB is a great environment to do this. However; be aware that many training samples are needed. And it may take a while to train.

(Slide 34) an alternative approach is called transfer learning, which can be fairly accurate in a smaller amount of time and with less data. So for the next example, we will see how to perform transfer learning to classify 5 new categories of objects.

**Demo #2:**

For this demo, Let's import a pre-trained CNN and re-train it to classify 5 different foods that you might find in a restaurant. I have my five different categories of food separated into 5 separate folders

So I can use imageDatastore to import that data, and label it based on those folder names

So you can see my 5 different labels of objects.

(type tbl.Label)

I first want to import my pre-trained CNN. I've already downloaded the CNN, so I'm going to simply load it from a saved file, but we do have a handy helper function to help import this CNN for the first time and convert it to the proper series network format.

Now we can take a look at the layers of this network. We can see that This CNN has already been trained to classify 1000 different objects, and we want to change this to instead classify our 5 different objects.

We can also look at the first layer, the input layer to see what type of data it is expecting. So we need to make sure our input data are these dimensions

Now we want to manipulate the CNN to perform our task. I'm going to take that last layer that is classifying 1000 categories, and switch that to 5.

I'm also going to alter the learning weights, so the last layer the one that I am manipulating, I want that to learn fastest, and I want to keep all of the other learning rates of the other layers the same, so all of the training that has already been accomplished in the network does not get altered as the final layer where I’m doing the 5 category classification.

Finally, before we re-train, I want to ensure that each class has the same number of input images and I once again want to set aside some images to validate the classifier.

Here is where the training happens: If we take a look at the documentation, we have a number of different parameters available to alter how the network

I'll point out that I altered the initial learning rate and the number of Epochs, or stages.

*(back to example code)*

For this example, I set the Epochs (the number of stages) to 20 and the InitialLearningRate to a smaller number than the default

You have lots of control over how the network is trained by changing these parameters, but because there are so many combinations of training options, it may take a few iterations to find the best combination to work the best with your data.

I'm going to run this and in the interest of time I've sped up the training to show what it looks like to train the entire network. So keep in mind that the actual time this took to train took much longer

While this is training you get information on how the network is performing. you can see that it is getting a very high accuracy in just a few stages, and while this might sound good it could be an indication that the network is overfitting the data.

Now that the training is done, we want to test to see how well it does on our training data. We set aside a number of test samples, and now we can use those to classify what category each of the images is predicted to be.

So for all of our data, we get an accuracy of 84 percent. We can look at the confusion matrix, very similar to the one in the classification learner app, which shows the accuracy of each class along the diagonal.

And the one where it's performing the worst, only 77% accurate is this 3rd category

(type tbl.Label)

And the third label is the 'hamburger' class.

So we can write a few lines of code to pull out the test images of the class we want to investigate. And we also limit the images to those that were misclassified. This can be a way of visualizing why the network is misidentifying this category.

And we can see that some of these images of hamburgers aren't very good. They're either blurry, bad lighting, some aren't hamburgers at all. So this is a good reminder to make sure your data that you are using to train and test is of high quality and represents how you hope the classifier will respond.

So going back to the very beginning of this script again, that's exactly what I did, I went back through all of the data and cut out the images that either weren't of high enough quality (low lighting, weird angles) and also weren't of the object we were trying to classify.

And this is a good reminder that before you get too deep into training a neural network or any classifier. Make sure the network isn't getting confused based on low quality images.

Another thing I changed for this second pass was to add another layer to the network. This is just another trick you can use to prevent overfitting of your data, and to add more non-linearity to the network.

I'm going to run this from the beginning again, but in the interest of time I'm not going to show training of the network.

This time when we test the Network, we get a result of over 90% So with those small changes, we can directly affect the accuracy of the results.

So in order to validate this CNN, I went out to restaurants and took video of food from those 5 categories I know what you're thinking: it's a tough job, but someone has to do it

And we can see as the video is streaming, how the classifier is working and also the confidence of that prediction.

Sadly, I'm in an office and not in a restaurant right now, but keep in mind that while this is pre-recorded, I could also run this on a live video stream as well.

**Demo #2 takeaways: (Slide 35)**

Here are two takeaways from the second demo:

Always use good data. Do this before spending lots of time training the network. If you are passing bad images into the network, expect the accuracy to be lower than you would like.

The second is to remember that transfer learning can be a very powerful technique to classify new objects. You have control over the network with a variety of training options. It may take a few times to get the right combination of training options to converge on a highly accurate solution, but MATLAB is the right environment to try these combinations out quickly.

Our final example is a combination of a deep learning a machine learning approach. So what does that look like? We have our deep learning workflow again but now we’re going to use deep elarning as a feature extractor, and pull those features into a traditional machine learning algorithm. So we’re taking advantage of the many layers of the CNN architecture and extracting out the most relevant features.

Let’s see this machine learning and deep learning combine approach in action.

**Demo #3:**

So as promised, we’re revisiting the first demo. We’re using the same categories as before, and we still want to identify what scene we’re in. This time using a combination deep and machine approach.

Some of this should look familiar by now. We’re still using image datastore to handle accessing our images, and we’re splitting each label into exactly the same number of images (904).

And we can look at a sample image from each of our 4 categories.

Once I’ve imported the pre-trained CNN, I can take a look at the layers. Pulling the features at the layer fc7 gives us the culmination of training from all of the previous layers, and we can look at the first layer to remind ourselves the input requirements.

I can specify my own custom read function using imageDatastore. This allows me to do any of my preprocessing of my data in one place. The custom read function would also be the place where I could handle reading in any non-standard image formats. But in this case, I’m simple ensuring an RGB image, and resize to fit the input size requirement.

So I once again want to separate my data into a training and a test set, and now I want to extract those features from the CNN.

The activations of the network at FC7 is the result of passing the image through all of the learned filters of the CNN. This is likely going to be a good feature extractor because the original CNN was trained on millions of images.

(Time in Webinar 23:45) So now we have those features. This is typically where I can pull up the Classification Learner App, and choose a classifier, but if you already have extracted code from the app or if you simply know the classifier you want to use and the function to call it you can simply call that directly in the script.

So now I have a linear SVM that was trained using the features from the CNN, and I want to evaluate how that classifier performs using the test data. I’m going to extract the same features from my test data and I’m going to predict using the classifier that was just created. As this is running, recall the accuracy of the first example (70%).

So in this case, we were able to considerably increase the accuracy of the prediction by using a deep learning approach. Finally, we can visually inspect how our classifier is performing by going through samples from our test set and comparing the prediction with the actual category.

**(Slide 43) That concludes the final demo for this webinar**

Before we finish, I want to start a discussion on deep learning vs. machine learning

You’ve seen 3 demos today which featured machine and deep learning techniques,

But how do you know which one to use?

With Machine Learning, you will have the many options to train on many different classifiers, and you have a wide range of feature extraction methods to choose. Also, if you understand your data, you may intuitively know which features are the best features that will produce good results.

Plus, you have the flexibility to choose a combination approach: different classifiers and different features and see which combination works best for your data. (MATLAB is a great tool to try these combinations quickly)

There is a lot of hype around deep learning, and for good reason. The accuracy of some models is very high. You don’t have to understand which features are best to represent in the model: these are chosen for you.

But in a deep learning model, it can take a while to train and because it is a black box, if something isn’t working correctly, it may be hard to figure out how to debug it.

I’ll leave you with the summary of the last demo which is the deep learning and machine learning combination approach

You can use deep learning as a powerful feature extractor and then have the ability to choose a classifier that gives you the most accurate results

But always remember there will not be a one-size fits all approach and be sure to try out many options for getting the best results for your particular application.

Thank you for tuning in today!

If you have any questions you can reach me and the rest of the computer vision and deep learning team

At image-processing@mathworks.com