1. **Which are the standard CNN architectures used for classification problems?**

https://vitalflux.com/different-types-of-cnn-architectures-explained-examples/

AlexNet – Designed for large-scale image datasets, 5 convolutional layers (3 fully connected, 2 dropout) with ReLu activation in all layers, softmax in output layer. 60 Million Parameters

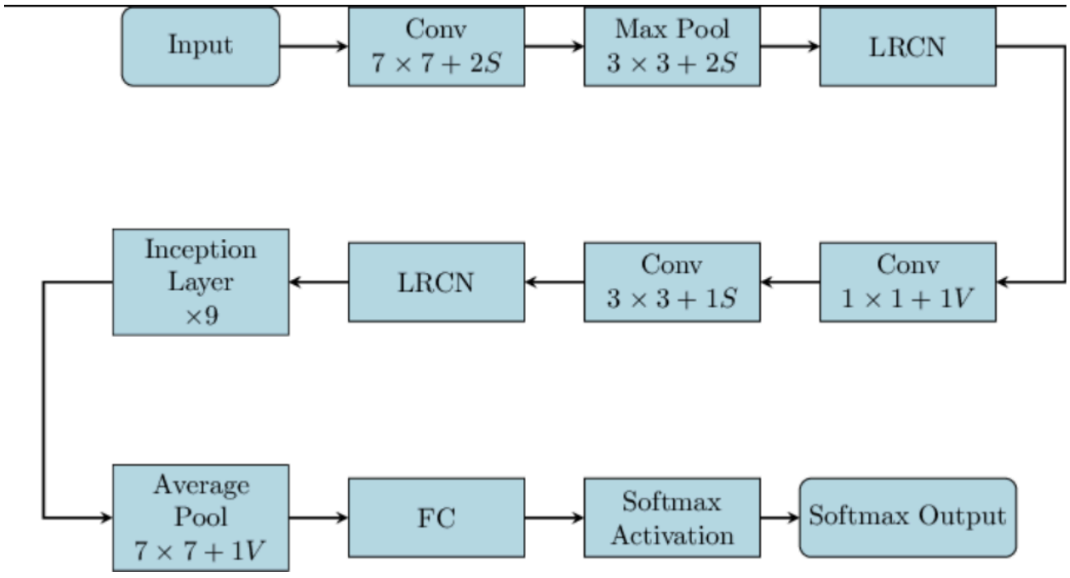
ZFNet – Has relatively fewer parameters than AlexNet and generally outperforms it, it achieved better performance on smaller datasets (~1000 images per class) so it may be better for our purposes which has a similar scale to 1000 per class, Expanded the size of the middle convolutional layers and made the stride (how many pixels the filter moves in each step during convolution) and filter size (number of pixels the filter looks at at one time) smaller on the first layer

> seven layers: Convolutional layer, max-pooling layer (downscaling), concatenation layer, convolutional layer with linear activation function, and stride one, dropout for regularization purposes applied before the fully connected output

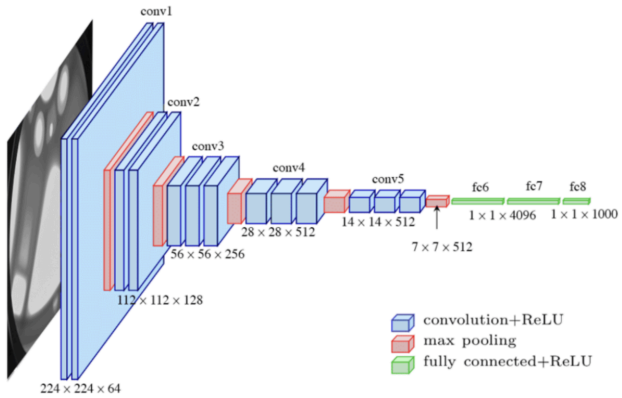
More computationally efficient than AlexNet by introducing an approximate inference stage through deconvolutional layers in the middle of the CNN

ResNet – 152 layers with over 1 Million Parameters, very deep even for CNN, ResNet includes Skip connections that solve the vanishing gradient problem (the gradient gets calculated at zero, and the model cannot train further because it cannot move) and allow for more layers and generally outperforming AlexNet

GoogLeNet – Much reduced error rate, Introduced 1x1 convolutional and global average pooling (instead of sliding across images, it looks at each pixel), VERY computationally expensive (uses heavy unpooling layers, shortcut connections between first two layers, adds filters in later layers)



VGGNet – trained on over 1 billion images in 1000 classes, very large filters (224X224 and 4096 convolutional features), requires a lot of data to train and doesn't perform well on smaller datasets, below is the standard VGGNet architecture



DenseNet – Implements dense connections to mitigate vanishing gradient, direct access to gradients from the loss function (promotes feature propagation), reduces feature reuse (from previous layers) which promotes efficiency by reducing redundancy, reduced parameters compared with ResNet (fewer parameters but still can be very deep without vanishing gradient)

Aspect	DenseNet	ResNet	VGG	Inception (GoogLeNet)	AlexNet
Connectivity	Dense connections	Shortcut connections	Sequential	Parallel paths	Sequential
Gradient Flow	Excellent	Good	Moderate	Good	Moderate
Parameter Efficiency	High	Moderate	Low	Moderate	Low
Feature Reuse	Extensive	Some	Minimal	Moderate	Minimal
Vanishing Gradient	Mitigated	Mitigated	Prone	Mitigated	Prone
Depth	Very deep, fewer parameters	Very deep	Deep, limited by training	Deep	Shallow compared to modern
Computational Cost	Moderate, higher memory usage	Moderate to high	High	Moderate	Moderate to high
Training Complexity	Moderate	Moderate to high	High	Moderate	Moderate
Performance	High, state-of-the-art	High, state-of-the-art	Good, but outperformed	High, competitive	Good for its time
Applications	Classification, detection, segmentation	Classification, detection, segmentation	Classification, feature extraction	Classification, detection, segmentation	Classification, early benchmarks
Introduced	2017	2015	2014	2015	2012

2. **Given our images are histopathology images (microscopic), which CNN architectures will be appropriate for our data?**

https://www.sciencedirect.com/science/article/abs/pii/S0933365719307110
This paper says **ResNet-50** and DenseNet-161
"DenseNet-161 tested on grayscale images and obtained the best classification accuracy of 97.89%. Additionally, ResNet-50 pre-trained model was tested on the color images of the Kimia Path24 dataset and achieved the highest classification accuracy of 98.87%."

This paper was looking at many different tissue types and classifying different pathogens within those tissues. It was a much broader study and not necessarily looking at the epithelium specifically, but it is using computer vision to classify sections of tissue, and has effective results.

https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-017-1685-x
This paper says **AlexNet, replace the final output layer of the CNN with Support Vector Machines** (SVM)
"According to our experiments, the framework proposed has shown state-of-the-art performance on a brain tumor dataset from the MICCAI 2014 Brain Tumor Digital Pathology Challenge and a colon cancer histopathology image dataset."
MICCAI challenge: "the target is to distinguish images of glioblastoma multiforme (GBM) and low grade glioma (LGG) cancer. The training set has 22 LGG images and 23 GBM images, and the testing set has 40 images." **97.5% accuracy**
Colon cancer "355 cancer and 362 normal images are used for binary tasks." **98.0% accuracy**

Made a point of saying this was large data because the images are so high-resolution
This paper dod both heat-mapping of cancer severity and binary classification – we are more interested in the binary classification, and it got good results with colon cancer in that method
It also outlines the improved results with image segmentation as opposed to whole-slide images

** All these papers are done on whole slide images, and they do the segmentation in a small grid without paying attention to the epithelium like how we are, and they are all done on H&E stains

My suggestion: We build a baseline ResNet50 and AlexNet model, then see which performs better initially
ResNet is more developed and successful in general classifying tasks, but the second paper is more applicable to our problem than the first and they had good results with AlexNet
** and try DenseNet-161 and ZFNet if we run into major issues

3. **Which python library, and functions will be use to train the models identified in (2)**

ResNet-50 – pytorch
https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/Classification/ConvNets/resnet50v1.5
Tensorflow/keras
https://medium.com/@nitishkundu1993/exploring-resnet50-an-in-depth-look-at-the-model-architecture-and-code-implementation-d8d8fa67e46f

AlexNet – pytorch
https://github.com/pytorch/vision/blob/main/torchvision/models/alexnet.py
Tensorflow/Keras
https://thecleverprogrammer.com/2021/12/13/alexnet-architecture-using-python/

DenseNet-161 – pytorch
https://pytorch.org/vision/main/models/generated/torchvision.models.densenet161.html

Pytorch V.S. Tensorflow

- Pytorch is more python-friendly, and ideal for research and rapid prototyping
- TensorFlow is a mature deep learning framework with strong visualization capabilities and several options for high-level model development. It has production-ready deployment options

4. **What hyperparameters will we need to tune?**

https://medium.com/@sathemayuri52/fine-tuning-convolutional-neural-networks-a-guide-to-hyperparameters-in-cnns-with-python-and-keras-a6cc52926ec2
https://medium.com/@sengupta.joy4u/how-to-decide-the-hyperparameters-in-cnn-bfa37b608046
https://ai.stackexchange.com/questions/17512/when-training-a-cnn-what-are-the-hyperparameters-to-tune-first

Model Hyperparameters

1. Filter/Kernel Size
    a. Size of filters used in each layer.
    b. Larger filter size means more computational cost, but captures more features of the data
2. Number of layers
    a. Depth of neural network, i.e. number of convolutional layers used in the model
    b. Deeper models require more computational power, but are generally more accurate

Training Hyperparameters

1. Learning Rate
    a. Step size used to update weights during training
    b. Larger means faster convergence or potential overshooting, and smaller means slower convergence but potentially more accurate
2. Batch size
    a. Number of samples that are processed with each iteration

    b. Larger batch reduces variance and improves stability, increases time and cost but can overfit.
3. Epochs
    a. Number of times you go through the entire training dataset

Regularization Hyperparameters

1. Dropout Size
    a. Randomly dropping neurons in a layer to avoid relying too much on specific neurons
    b. Makes the model more robust and generalizable because you can't rely on one neuron to make most of the predictions
2. Weight regularization
    a. Avoid making the weights too large of a size

There are many hyperparameters we can and likely should tune. However, hyperparameter tuning is largely an optimization problem. It is very expensive to go through the training process and then also validate our model based on our data. Thus, especially given our limited resources, it would be best to look at which hyperparameters are more important to look at.

https://www.cell.com/heliyon/fulltext/S2405-8440(24)02617-3
This paper looked at many different types of hyperparameter tuning methods and various hyperparameters in a CNN model and quantified the importance of certain hyperparameters:

Table 4 The fANOVA-based parameter importance without stratification per dataset.

Hyperparameters	Parameter importance
Learning rate	0.187
Input image size	0.171
Dropout	0.138
Number of fine-tuned layers	0.037
Batch size	0.029
GD optimization technique	0.025

They also used different datasets to look at what the top hyperparameters would be for each one, and the top two were input image size and learning rate.

	The most important parameter pairs	
Dataset	**Set I**	**Set II**
Stanford Dogs	Optimization method & Input image size	Number of fine-tuned layers & Input image size
CIFAR-100	Optimization method & Number of fine-tuned layers	Number of fine-tuned layers & Input image size
MIO-TCD*	Optimization method & Dropout	and Dropout & Number of fine-tuned layers
All datasets	Learning rate & Input image size	Learning rate & Dropout

The importance of each hyperparameter is dependent on the dataset being used, so it would be useful to determine which hyperparameters will be most effective for us to focus on so we don't waste computational resources tuning parameters that will have a minimal effect on our accuracy.

The study also found that training on subsets of the training data can be just as effective as training on the whole dataset, which is also helpful given our limited computational resources.

Hyperparameter optimization dataset	**Learning rate**	**Dropout**	**Optimization method**	**Number of fine-tuned layers**	**Accuracy on the test set**
Balanced	0.0249	0.1941	Adam	132	**85%**
imbalanced	0.0103	0.3861	RMSprop	14	76%
imbalanced & class weighting	0.0158	0.1856	Adam	132	**86%**
imbalanced & augmentation	0.0014	0.2837	RMSprop	132	83%
imbalanced & class weighting & augmentation	0.0056	0.3621	Adam	132	**85%**
whole training set	0.0162	0.2333	Adadelta	132	81%

Study concluded that ASHA and Bayesian optimization techniques are better in classification accuracy than random and gridsearch, and required fewer iterations.

https://ieeexplore.ieee.org/abstract/document/9497988
This paper explores avenues to take when the CNN model is under/overfitting based on validation and training accuracy. Underfitting is fixed easily enough by increasing layers or the neurons per layer. Overfitting can be fixed through regularization techniques such as dropout,

batch normalization, or pruning some of the parameters. They describe different pruning methods.

5. **Are there any other deep learning models (other than CNN) that will be appropriate for our problem? You may ask health people and lit surveyors to share papers relevant to our problem.**

https://www.mdpi.com/2076-3417/13/9/5521
This article compares vision transformers and CNN for image classification. Transformers are primarily used for NLP tasks, and can be scaled and are extremely efficient. Breaks the initial image into 2d patches and then feeds them as one linear input into the model. Falls short of specialized CNN architectures (which would be in our case).
Pros:

- Robust when provided with large datasets
- Can adapt well to different input sizes
- Potential to reduce bias in anomaly detection
- Very fast and parallelizable

Cons:

- Doesn't generalize well when given small training dataset
- Requires large amount of fine-tuning to become very accurate
- Not as widely used so not many prior studies or benchmarks to go off of
- Computationally expensive when model increases in size or image size increases

https://proceedings.neurips.cc/paper_files/paper/2021/file/cba0a4ee5ccd02fda0fe3f9a3e7b89fe-Paper.pdf
MLP-Mixer - new type of model that came out based not on attention or convolution like transformers or CNNs. Based on multi-layer perceptrons. Simpler than models like vision transformers and CNN but attains competitive results. (Falls short of specialized CNN architectures, like our case)
Pros:

- Simpler than Vision Transformers or CNN's
- Just as efficient and competitive as CNN's and transformers
- Good for cases where computation resources are limited

Cons:

- Not as powerful as the other techniques for prediction accuracy

https://ieeexplore.ieee.org/abstract/document/9175815
https://www.researchgate.net/profile/T-Vijayakumar-2/publication/336466044_COMPARATIVE_STUDY_OF_CAPSULE_NEURAL_NETWORK_IN_VARIOUS_APPLICATIONS/links/6070127aa6fdcc5f7790b004/COMPARATIVE-STUDY-OF-CAPSULE-NEURAL-NETWORK-IN-VARIOUS-APPLICATIONS.pdf?_sg%5B0%5D=started_experiment_milestone&origin=journalDetail
Capsule Network - More robust to rotation and transformations in the images. Meant to handle cases where there isn't as large an amount of training data as you would want for special cases

in the data where there are transformations in the input images. Also handles the loss of information in spatial details such as rotation and location that you would get for CNNs.
Pros:

- Competitive, if not better accuracy on existing, popular training datasets compared to other capsule networks
- Robustness to transformations in the images
- On small datasets it performs better than CNNs
- Interpretable due to looking at activation vectors

Cons:

- Computationally much more expensive compared to regular CNNs
- Not as effective with large datasets and complex classification

6. **In what format we'll provide the input? Will they be png files, or will we convert the images into tabular data?**

Based off of nearly all the articles I've read about CNN, the images are typically fed as matrices with three channels for rgb values.
https://ieeexplore.ieee.org/abstract/document/7808140
This article fully dives into the preprocessing methods for images in use for CNNs. However, even in this the data was fed into the model as 3x32x32 matrices.
http://www.jurnal.iaii.or.id/index.php/RESTI/article/view/5417/886
This article as well as the one above do show that preprocessing the images is helpful in increasing model accuracy. This could be things such as normalizing the images, contrast enhancement, noise reduction, and light reduction (while the images are in matrix form).