Cervix Type Classification for Cancer Treatment with convolutional neural networks

By: TheDoctors

Iuliia Klykova
Emmanuel Ojuba
Victoria Gianello

Problem description

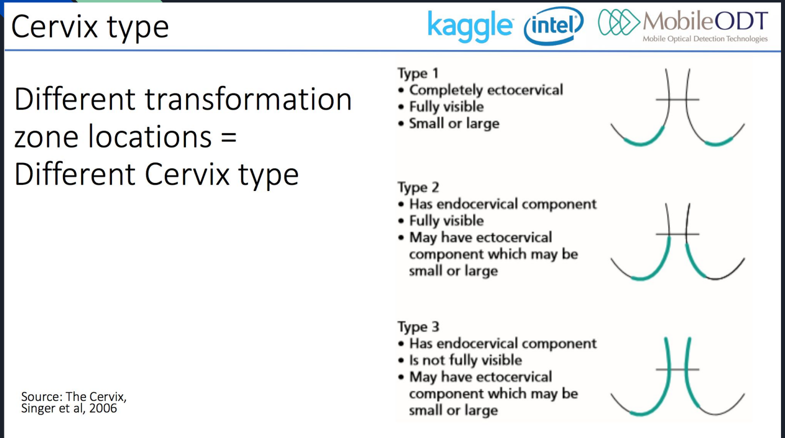


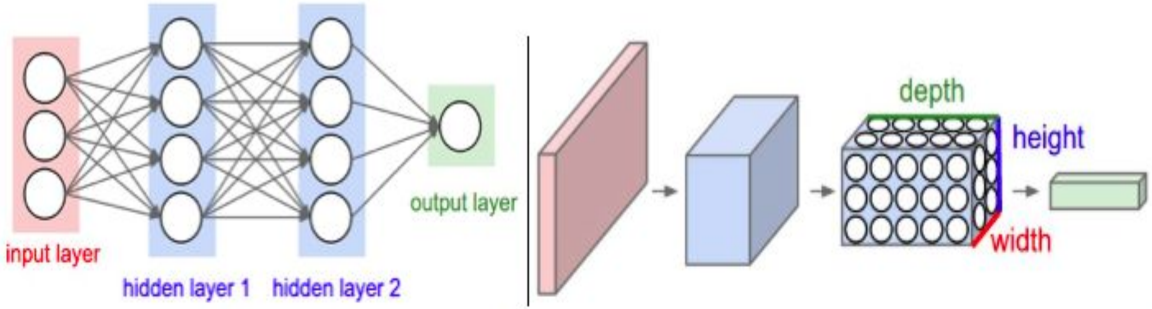
Figure 2. Example images of each cervix type. Image (a) is Type 1, image (b) is Type 2, and Image (c) is Type 3

Cervical cancer is easy to prevent in its precancerous stage. However, many times doctors are not able to give patients the appropriate treatment for their cervix anatomy. Therefore, we created a Neural Network that given an image of a cervix, classifies cervix types accurately. This will allow doctors give the right treatment more reliably.

Initial Training DataSet ~ 1500 images

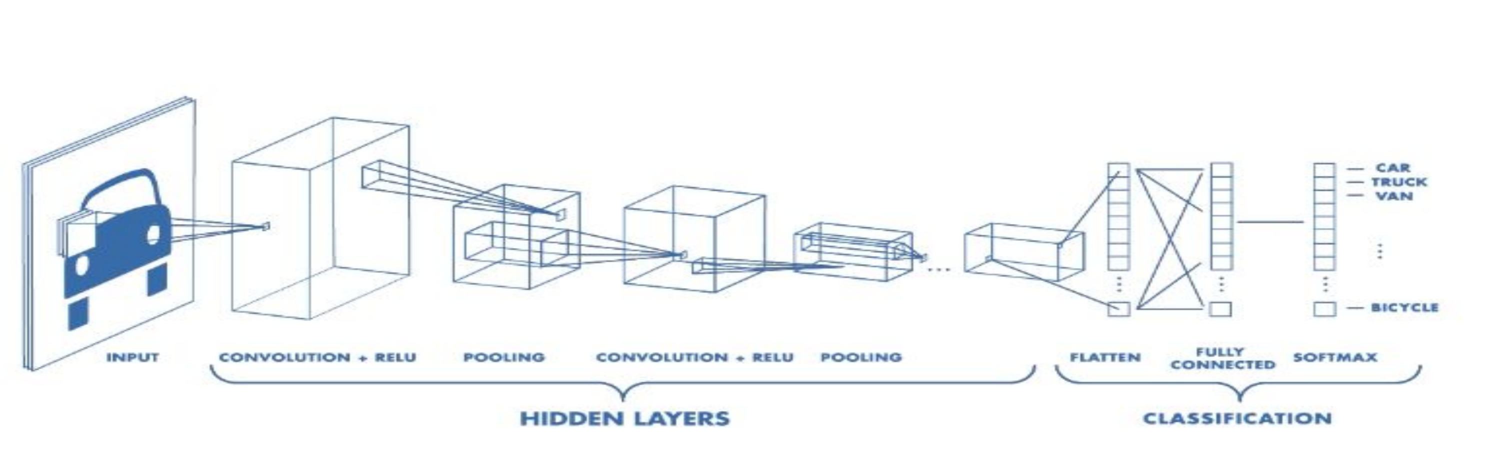
Type 1	Type 2	Type 3
250	781	450
17%	53%	30%

Convolutional Neural Networks



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

Convolutional Neural Networks



- Convolutional Layer: nxnxd filters which pass through the image, and compute various "activations"
- ReLU: Adds non-linearity to model, with reduced problem of 'vanishing gradients'
- Pooling: Downsample layer outputs to reduce computational load and overfitting (common methods include Maxpool, average pool)
- Flatten: Converts Output of final convolutional layer to 1-D Tensor (for fully connected layer).
- Global Average Pooling: Reduces computational overload of Flatten by averaging the output of final convolution (e.g 20x20x15 -> 1x1x15)
- Fully-Connected: Perceptron to every pixel in output of final convolution
- Softmax: Multi-class generalization of the the sigmoid non-linearity function
- Output

Prior Work

Payette, Rachleff, Van de Graaf (2018). *Intel and MobileODT Cervical Cancer Screening Kaggle Competition: Cervix Type Classification Using Deep Learning and Image Classification, 2018*

- 32 Layer Residual CNN
- 32 Layer Residual CNN with Dropout and Augmentation
- 32 Layer Residual CNN with Augmentation
- 32 Layer Residual CNN with Dropout
- Validation Accuracy: 0.56 - 0.58
- Test Cross-Entropy Loss: 0.87 - 0.923

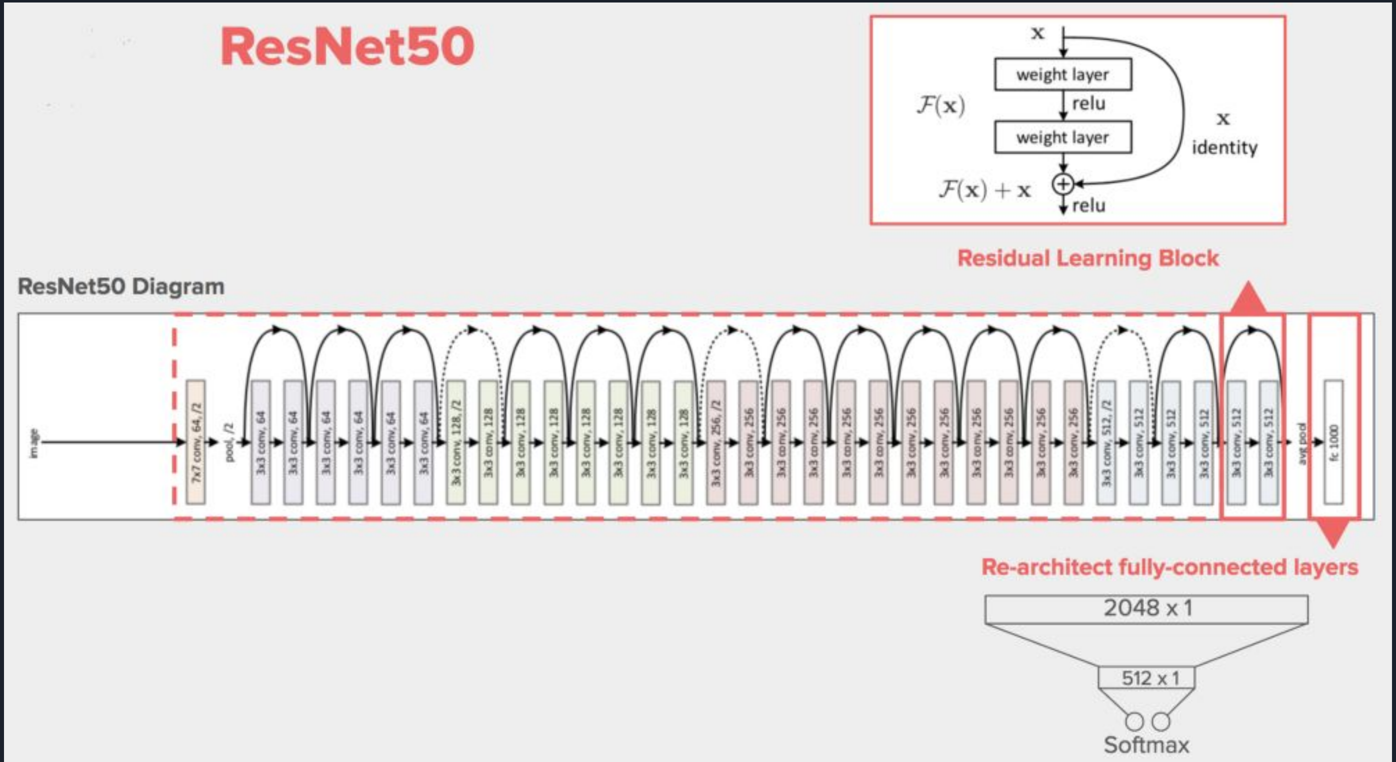
Kaggle Leader Board

#	△pub	Team Name	Kernel	Team Members	Score	Entries	Last
1	▲125	Towards Empirically Stable Tr...			0.76963	2	1y
2	▲114	I Rustandi			0.80277	8	1y
3	▲79	GRXJ			0.80829	5	1y
4	▲94	Ruslan Baikulov			0.81808	4	1y
5	▲73	BMCI			0.82205	5	1y
6	▲80	kubilai			0.82495	26	1y
7	▲124	Shai			0.82849	2	1y
8	▲93	Ryan Munion			0.83130	3	1y
9	▲78	ZFTurbo			0.83208	2	1y
10	▲134	Alexander Popov			0.83333	19	1y
11	▲136	mokp			0.83367	2	1y

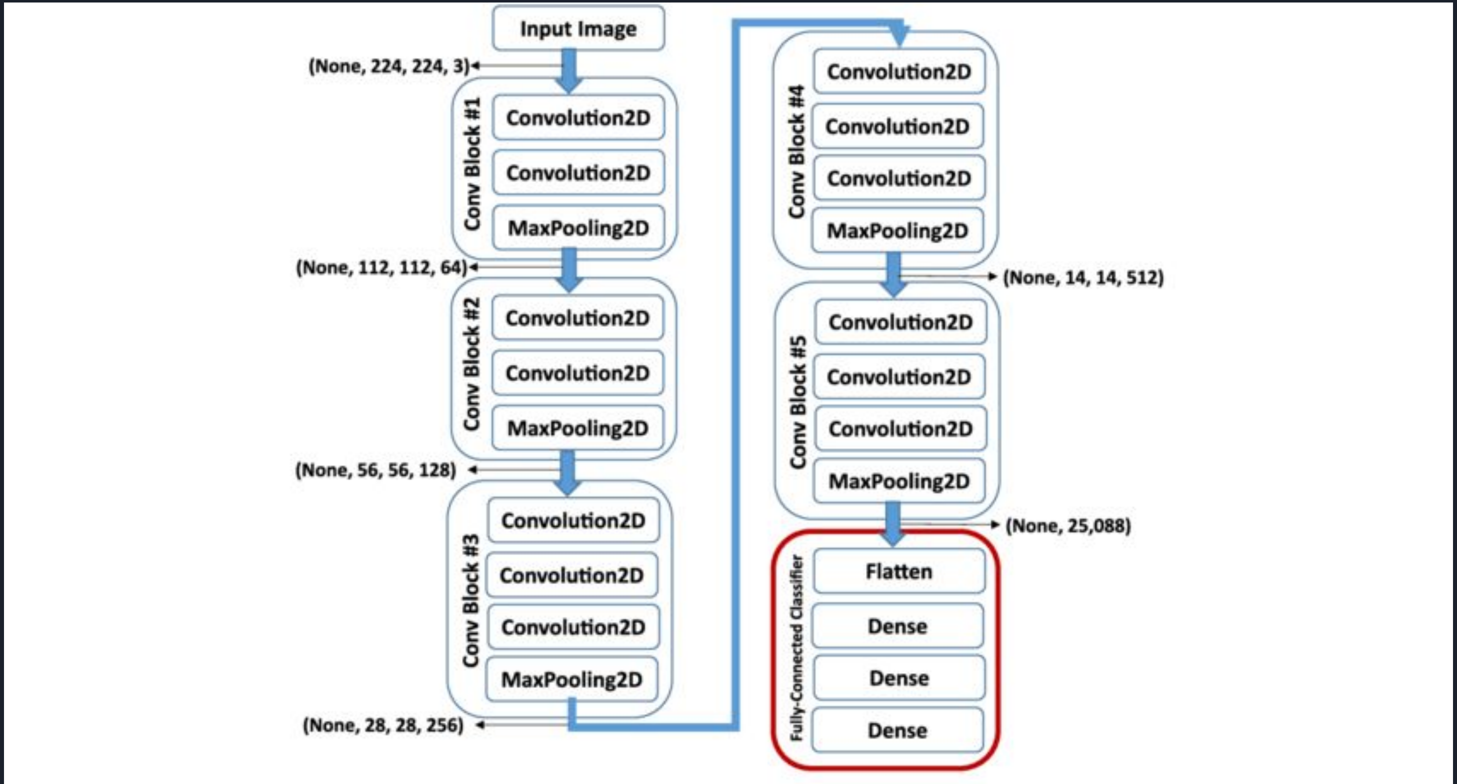
Model	Val Loss	Val Accuracy	Val F1 Score	Test Loss
32 Layer Residual CNN	.948	.580	.573	0.923
32 Layer Residual CNN with Dropout	.948	.564	.567	-
32 Layer Residual CNN with Dropout and Data Augmentation	.936	.576	.551	-
32 Layer Residual CNN with Data Augmentation	.879	.588	.578	.873

Steps followed to solve problem

ResNet50 diagram



VGG16 Inspired Model



VGG16 Inspired Model. Keras implementation

```
nClasses = 3
model = Sequential()
model.add(ZeroPadding2D((1, 1), input_shape=(64, 64, 3)))
model.add(Conv2D(64, kernel_size=3, strides = 1, activation='relu',
input_shape=input_shape))
model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(128, (3, 3),  activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(512, (3, 3),  activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(512, (3, 3),  activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(nClasses, activation='softmax'))
model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
```

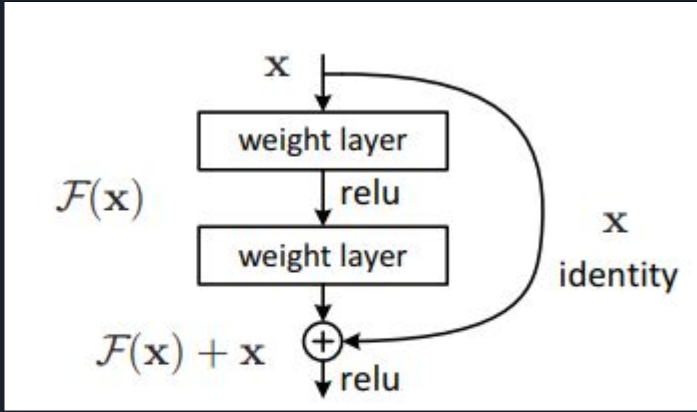
Model selection and implementation

ResNet 50 Model

- Uses batch normalization
- Learns residuals and uses them to get an optimized output

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

$$y^{(k)} = \gamma^{(k)} \widehat{x}^{(k)} + \beta^{(k)}.$$



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1 \ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i$ // mini-batch mean

$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2$ // mini-batch variance

$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$ // normalize

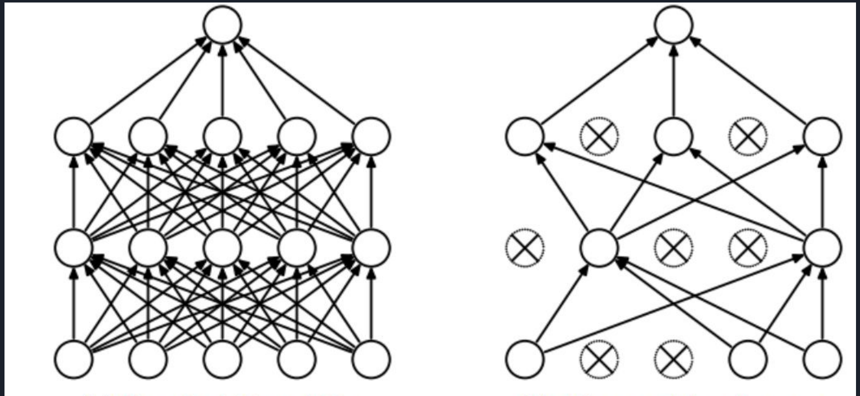
$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$ // scale and shift

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Dropout Regularization

Keras Implementation:

```
keras.layers.Dropout(rate, noise_shape=None, seed=None)
```



(a) Standard Neural Net (b) After applying dropout.

Training and testing

Metrics

- Accuracy: Percentage of Correct Predictions
- Multi-class Cross-Entropy Loss:

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\ln(p_{ij})$$

Train-Validation Split

- 75% - 25%
- Plain, randomized split

Optimization

Improve training time

- Adam optimizer - Kingma & Ba (2015)
    - Improvement of the SGD optimizer and momentum optimizer
    - Computes first and second moments
    - Proven to converge faster than other optimizers and very widely used

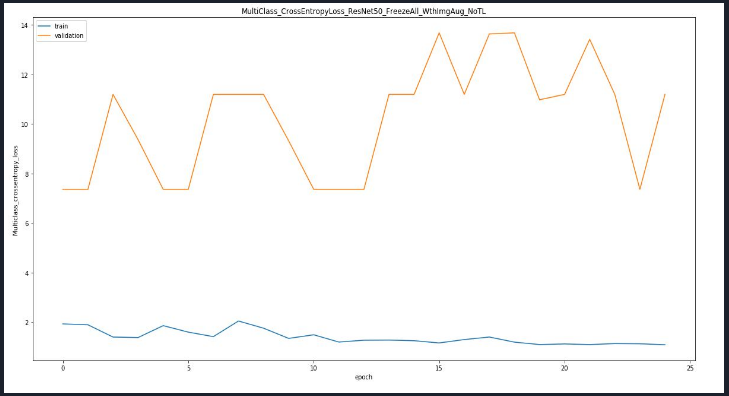
Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Results

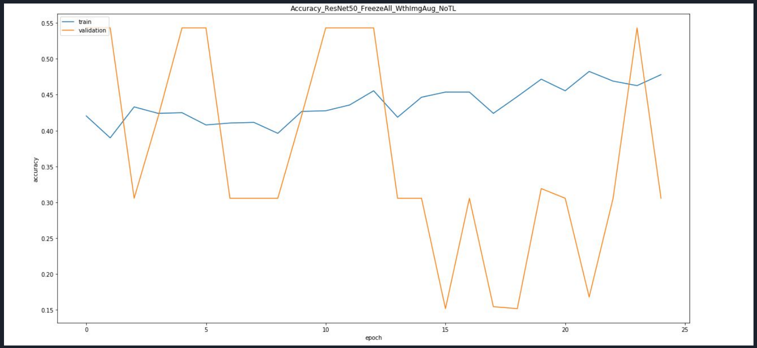
No Transfer Learning

- Unpredictable and unstable behavior on validation data
- Validation multi-class Loss hovering around 8 - 12
- Validation accuracy 0.15 - 0.55

Multi-Class Entropy Loss

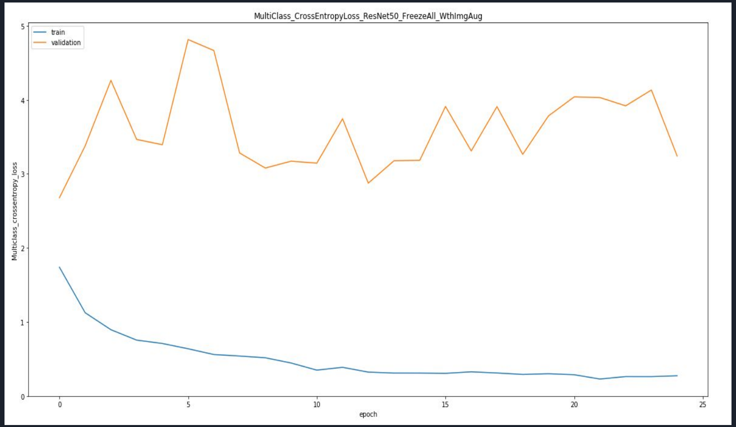


Accuracy

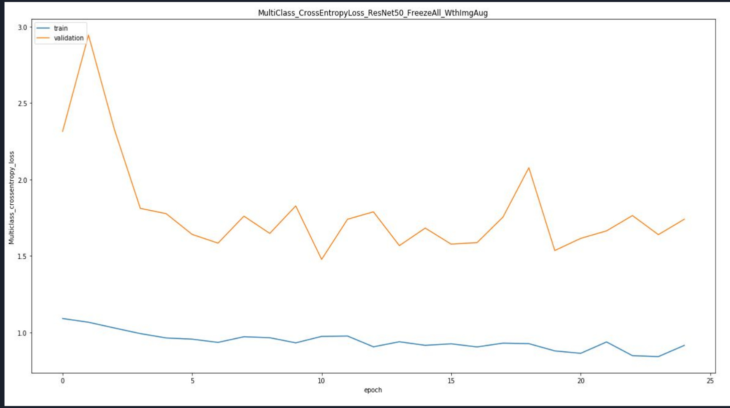


Results - ResNet50

Multi-Class Entropy Loss No Augmentation



Multi-Class Entropy Loss With Augmentation

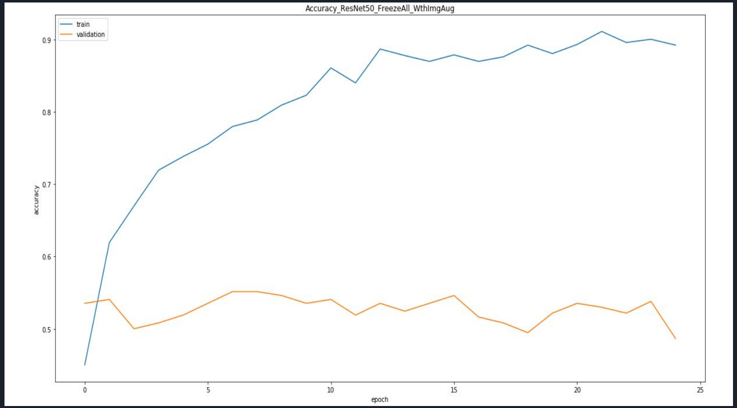


Loss:
V: 3 - 4
T ~ 0.1

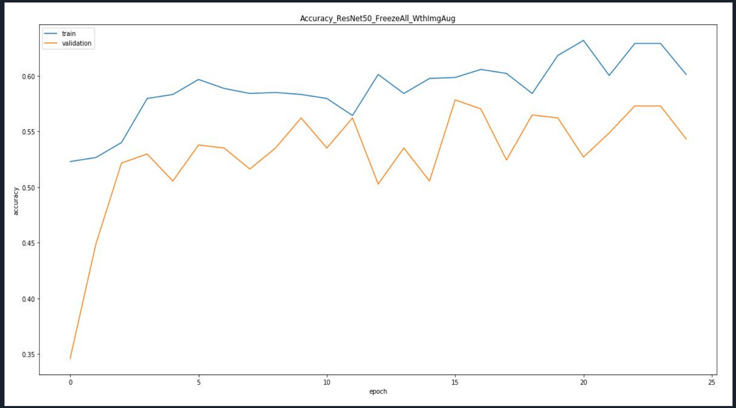
Loss:

V ~1.5
T ~ 1

Accuracy No Augmentation



Accuracy With Augmentation



Acc:

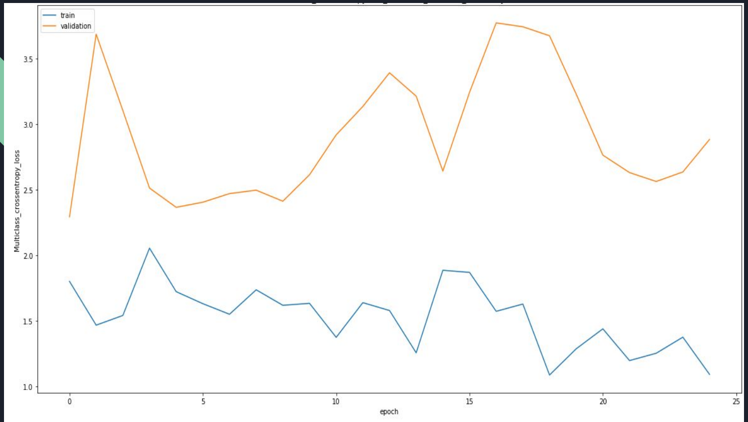
V: 0.5 -
0.55
T~ 0.9!!

Acc:

V0.50 -
0.55
T~0.6

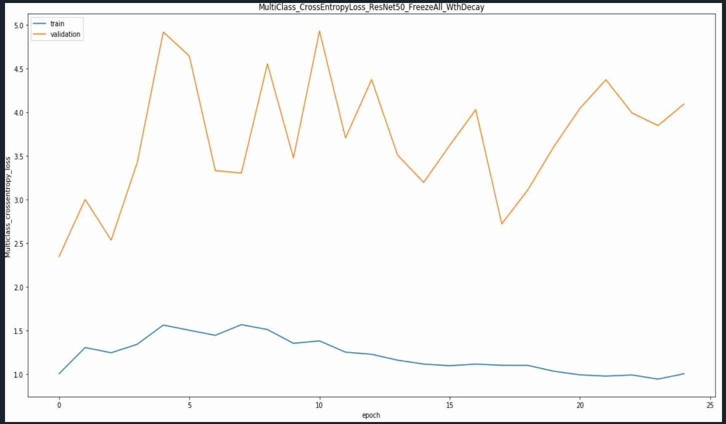
Results - ResNet50

Multi-Class Entropy Loss Fixed Learning rate



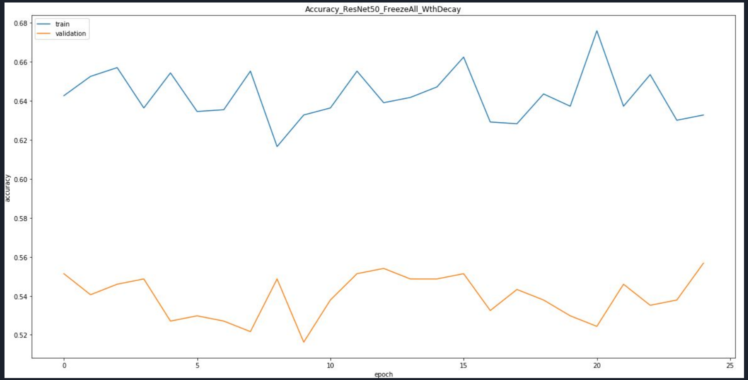
Loss:
T: ~1.5
V: 2.2 - 3.6

Multi-Class Entropy Loss Decaying Learning rate



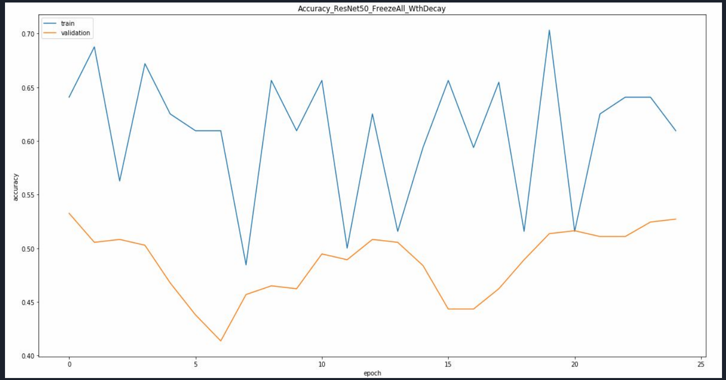
Loss:
T: ~1.2
V: 4

Accuracy Fixed Learning rate



Acc:
T: ~0.64
V: ~0.54

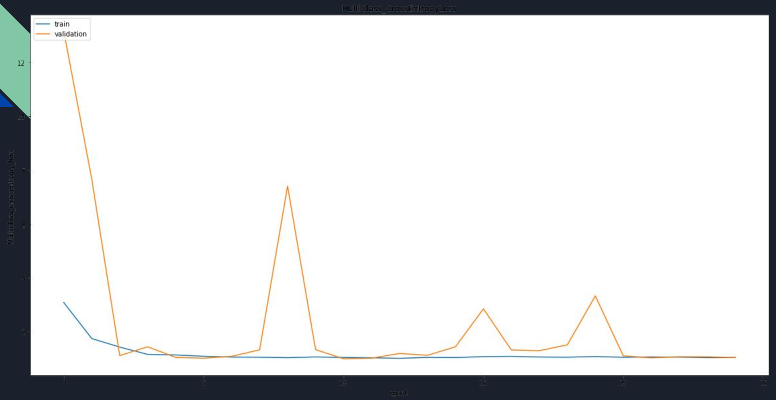
Accuracy Decaying Learning rate



Acc:
T: 0.55 - 0.7
V: 0.45 - 0.55

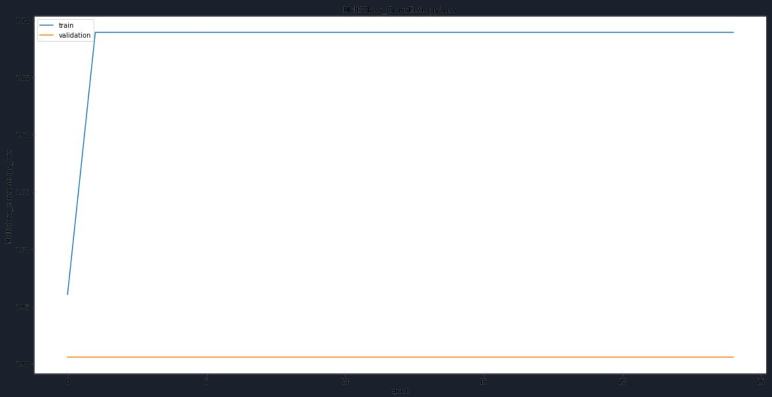
Results - VGG16 Inspired

Multi-Class Entropy Loss w/ Batch Normalization & Dropout



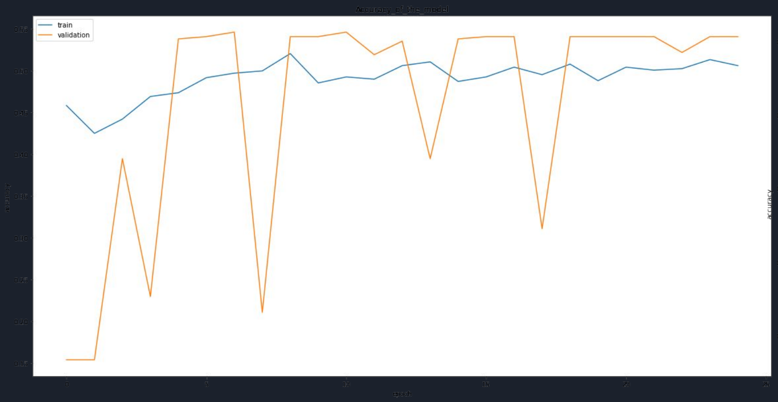
Loss:
T: <1.0
V: <1.0

Multi-Class Entropy Loss without Batch Normalization & Dropout



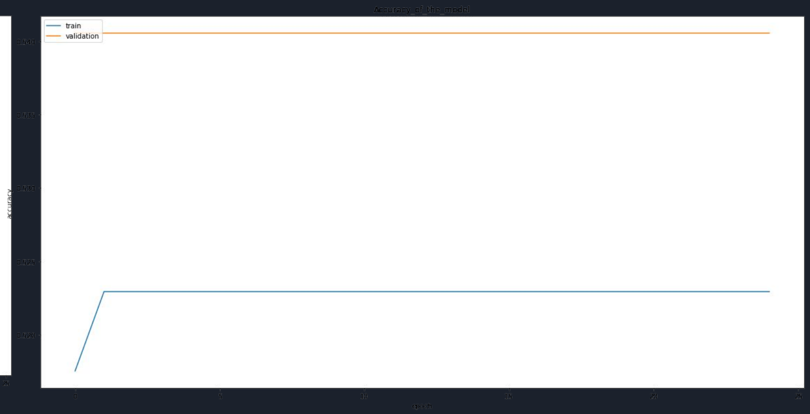
Loss:
T: ~7.50
V: ~7.41

Accuracy w/ Batch Normalization & Dropout



Acc:
T: ~0.50
V: ~0.55

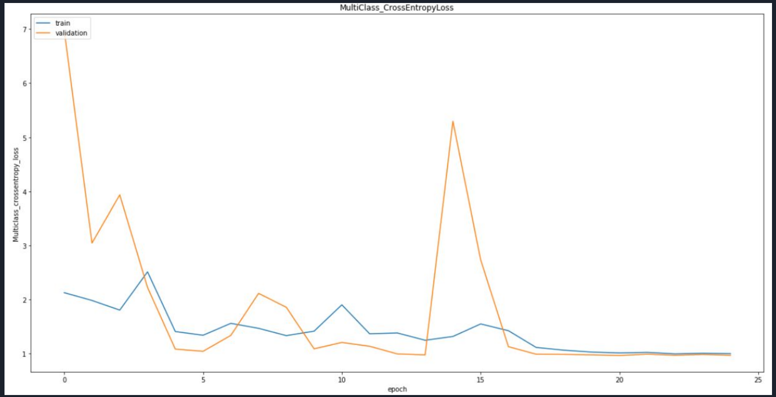
Accuracy without Batch Normalization & Dropout



Acc:
T: ~0.52
V: ~0.54

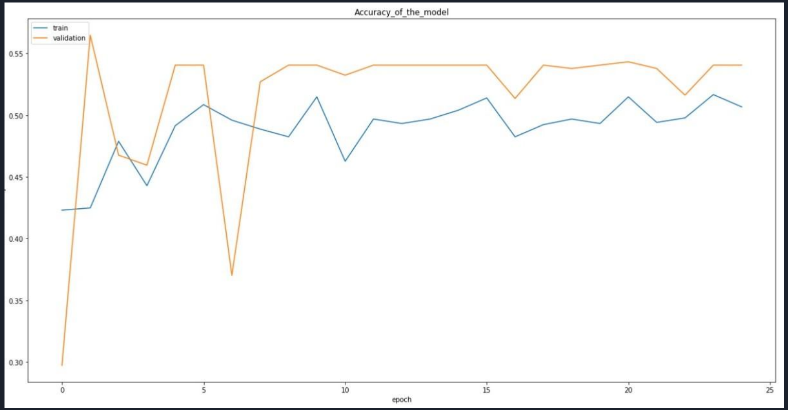
Results - VGG16 Inspired

Multi-Class Entropy Loss w/ Batch Normalization & no Dropout



Loss:
T: <1.0
V: <1.0

Accuracy w/ Batch Normalization & no Dropout



Acc:
T: ~0.50
V: ~0.54

When you spend hours training your neural network and it finally gives you an accuracy of 37%

Possible future enhancements

- Training with More Data (additional 4633 images were provided)
- Higher-Image Resolution: 224 x 224 vs 64 x 64
- Exploration of other Models (InceptionV3, Custom-built models)
- Ensembling of Models (Bagging, Boosting, Stacking)
- Enforcement of a more balanced class, confusion matrix could help with this

Truth

Predicted	Asphalt	Concrete	Grass	Tree	Building	Total
Asphalt	2385	4	0	1	4	2394
Concrete	0	332	0	0	1	333
Grass	0	1	908	8	0	917
Tree	0	0	0	1084	9	1093
Building	12	0	0	6	2053	2071
Total	2397	337	908	1099	2067	6808

- Data augmentation: Payette, Rachleff, Van de Graaf (2018) utilize a method of statistical cropping. Sharpening. Embossing
- Is the problem entirely solvable by a machine?

Thank you! Questions?

Sources

1. https://www.kaggle.com/c/intel-mobileodt-cervical-cancer-screening
2. https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050
3. Payette, Rachleff, Van de Graaf (2018). *Intel and MobileODT Cervical Cancer Screening Kaggle Competition: Cervix Type Classification Using Deep Learning and Image Classification.* http://cs231n.stanford.edu/reports/2017/pdfs/300.pdf
4. Greenspan, Hayit & Gordon, Shiri & Zimmerman, Gali & Lotenberg, Shelly & Jeronimo, Jose & Antani, Sameer & Long, L. (2009). Automatic Detection of Anatomical Landmarks in Uterine Cervix Images. IEEE transactions on medical imaging. 28. 454-68. 10.1109/TMI.2008.2007823.
5. Kingma & Ba (2015). Adam: A Method for Stochastic Optimization. https://arxiv.org/pdf/1412.6980.pdf
6. Ioffe & Szegedy (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift **https://arxiv.org/pdf/1502.03167.pdf**
7. He, Zhang, Ren & Sun (2015). Deep Residual Learning for Image Recognition **https://arxiv.org/pdf/1512.03385.pdf**
8. **http://slazebni.cs.illinois.edu/spring17/lec01_cnn_architectures.pdf**
9. **https://towardsdatascience.com/neural-network-architectures-156e5bad51ba**
10. **https://www.harrisgeospatial.com/docs/ENVIConfusionMatrix__ConfusionMatrix.html**