Bayesian Convolutional Neural Network with Variational Inference for Image Recognition

Yu Zhu

Department of Statistical Science

University of California, Santa Cruz

Outline

- Introduction to Convolutional Neural Network (CNN)

- Bayesian Convolutional Neural Network (BayesCNN):
  - Motivations
  - Construction

- Bayes by Backprop:
  - Cost function
  - Optimization algorithm
  - Spike-and-Slab prior
  - Minibatches and KL re-weighting
  - Advantages and disadvantages

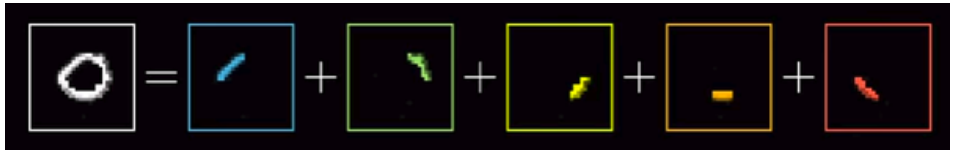
- Uncertainty Quantification

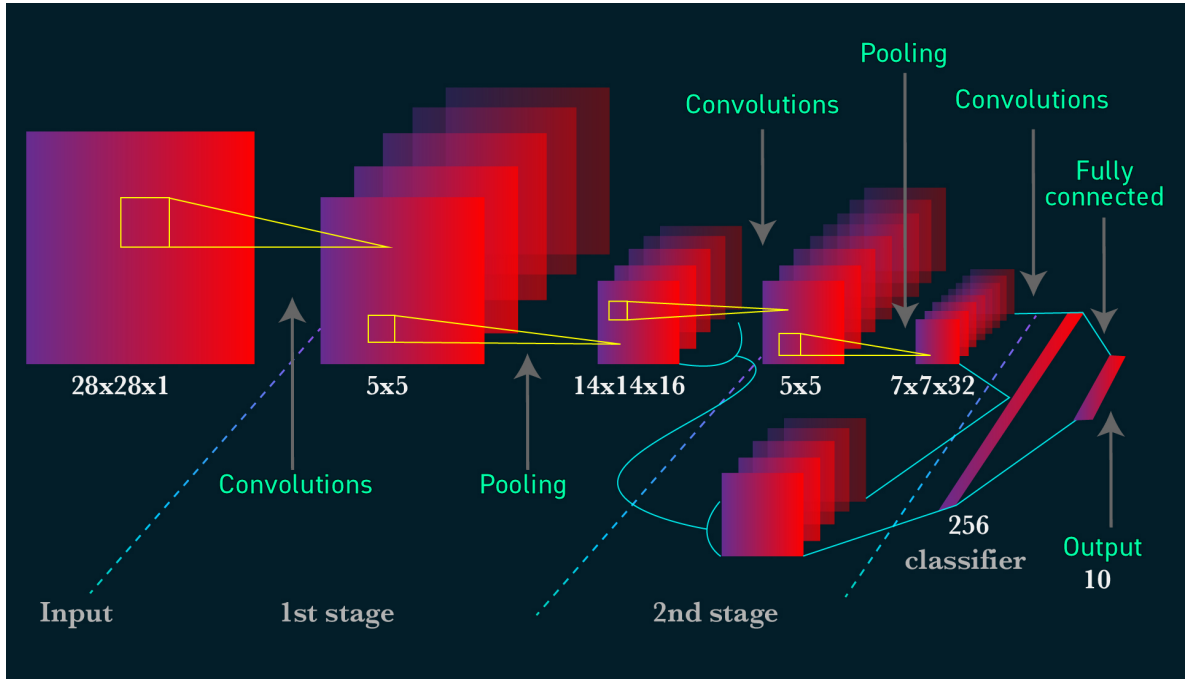
- Case Study: Simpson Characters Recognition
  - Data description
  - CNN structure and model setting
  - Accuracy comparisons
  - Posterior predictive probability credible intervals

Convolutional Neural Network (CNN)

- The unstructured image data:
  - An image usually has a resolution of n1 × n2 pixels, with three eight-bit numbers per pixel representing red, green and blue (Eg, CIFAR100 data, each image is under the size of 32 × 32 pixels).
  - The numbers for each image are organized in a 3-D matrix called a *feature map*, where the size of the first two dimensions corresponds to the length and width of the images in pixels, and the third is the channel axis with dimension equal to 3, representing the three colors (eg, white as (255, 255, 255)).

- Motivation for CNN in image classification:
  - Computational efficiency: the parameter space can be extremely large for the image data. The number of parameters in a neural network grows rapidly with the increase in the number of layers. CNNs are very effective in reducing the number of parameters without losing on the quality of models
  - CNNs can better identify the edges of objects in any image



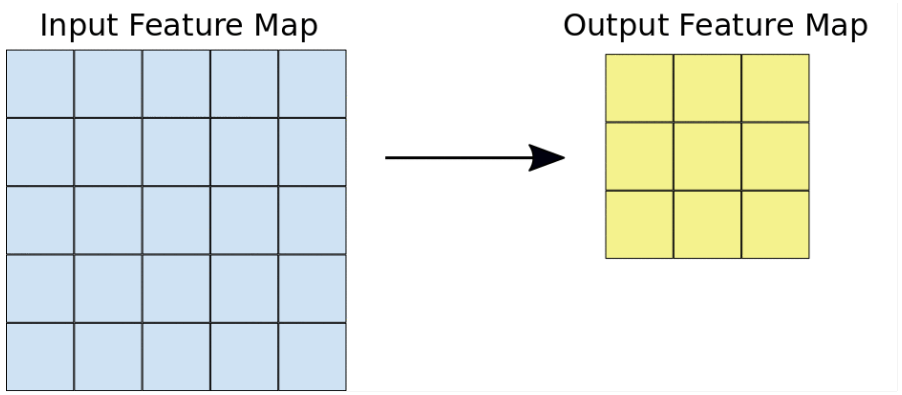


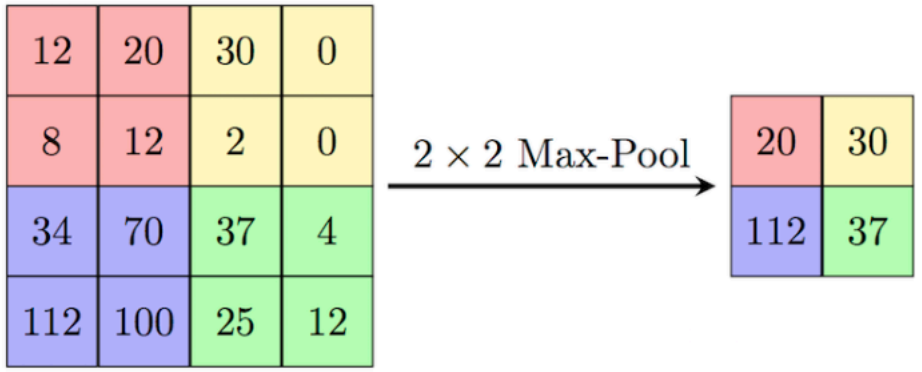
Convolutional Neural Network (CNN)

- Convolution:
  - A *convolution* extracts tiles of the input feature map, and applies filters to them to compute new features, producing an output feature map, or *convolved feature* (which may have a different size and depth than the input feature map).

- Training phases:
  - ***Feedforward***: obtain the predicted probability of each class with information moving from the input layer through hidden layers to the output layer in a forward direction.
  - ***Backproporgation***: find optimal values of parameters in order to minimize the loss function L. Specifically, deriving the gradient of parameters in a backward direction under some optimization algorithm and update the parameter.

- We need to update the fixed weights in the learning layers such as CONV layers and the fully-connected layer





Bayesian Convolutional Neural Network (BayesCNN)

- Existed challenges for CNN:
  - CNN tends to overfitting on small dataset, making over-confident decisions
    We usually need to apply some regularization techniques such as early stopping, weight decay, L1/L2 regularizations and dropout

- Motivation for BayesCNN:
  - Reduce the chances of overfitting especially while training on small-size data
  - Uncertainty estimation: BayesCNN is capable of measuring the uncertainty in the prediction
  - Regularization: using a prior probability distribution to integrate out the parameters, the average is computed across many models during training, which gives a regularization effect to the network

- BayesCNN construction:
  - Apply probability distributions on the parameters, learn the posterior distribution of parameters and infer model posterior
  - Usually use variational inference approaches to approximate the model posterior
    - Use Gaussian distributions for Bayesian NN posterior approximation (Blundell et al. (2015))
    - A gradient-based optimization procedure on the dropout neural network, equivalent to a specific variational approximation on a Bayesian neural network (Gal 2016)
    - Similar optimization procedure on batch normalization layers (Teye et al. 2018)

Bayes by Backprop

- Bayes by Backprop (Blundell et al, 2015):
  - A variational inference method
  - Approximate the posterior distribution by a variational distribution $q_\theta(\boldsymbol{\omega})$ indexed by variational parameter $\theta \in \Theta$
  - Optimize the variational distribution via minimizing the closeness between $q_\theta(\boldsymbol{\omega})$ and $\mathrm{p}(\boldsymbol{\omega}|D)$ measured by KL divergence
  - Find the optimal parameter θ that

$$\theta^* = {}^{argmin}_{\theta} KL\{q_\theta(\boldsymbol{\omega}) \parallel p(\boldsymbol{\omega}|D)\}$$

$$\text{where the } KL\{q_\theta(\boldsymbol{\omega}) \parallel p(\boldsymbol{\omega}|D)\} \coloneqq \int q_\theta(\boldsymbol{\omega})\, log \frac{q_\theta(\boldsymbol{\omega})}{p(\boldsymbol{\omega}|D)}\, d\boldsymbol{\omega}$$

  - The resulting cost function is known as the variational free enery or the expected lower bound (ELBO)

$$F(D,\theta) = KL\{q_\theta(\boldsymbol{\omega}) \parallel p(\boldsymbol{\omega})\} - \int q_\theta(\boldsymbol{\omega})\, logp(D|\boldsymbol{\omega}) d\boldsymbol{\omega}$$

$$= KL\{q_\theta(\boldsymbol{\omega}) \parallel p(\boldsymbol{\omega})\} - \mathbb{E}_{q_\theta(\boldsymbol{\omega})}[logp(D|\boldsymbol{\omega})]$$

$$= \mathbb{E}_{q_\theta(\boldsymbol{\omega})}[log \frac{q_\theta(\boldsymbol{\omega})}{p(\boldsymbol{\omega})}] - \mathbb{E}_{q_\theta(\boldsymbol{\omega})}[logp(D|\boldsymbol{\omega})]$$

$$= \mathbb{E}_{q_\theta(\boldsymbol{\omega})}\left[log \frac{q_\theta(\boldsymbol{\omega})}{p(\boldsymbol{\omega})} - logp(D|\boldsymbol{\omega})\right]$$

$$= \mathbb{E}_{q_\theta(\boldsymbol{\omega})}[logq_\theta(\boldsymbol{\omega}) - logp(\boldsymbol{\omega}) - logp(D|\boldsymbol{\omega})]$$

  - The cost function embodies a trade-off between satisfying the complexity of the data D and satisfying the simplicity prior $p(\boldsymbol{\omega})$

Bayes by Backprop

- Unbiased Monte Carlo gradients:
  - Under certain conditions, the derivative of an expectation can be expressed as the expectation of a derivative:

Proposition 1. Let ϵ be a random variable having a probability density given by $q(\epsilon)$ and let $\boldsymbol{\omega} = t(\theta, \epsilon)$ where $t(\theta, \epsilon)$ is a deterministic function. Suppose further that the marginal probability density of $\boldsymbol{\omega}$, $q_\theta(\boldsymbol{\omega})$, is such that $q(\epsilon)d\epsilon = q_\theta(\boldsymbol{\omega})d\boldsymbol{\omega}$. Then for a function f with derivatives in $\boldsymbol{\omega}$:

$$\frac{\partial}{\partial \theta}\mathbb{E}_{q_\theta(\boldsymbol{\omega})}[f(\boldsymbol{\omega}, \theta)] = \mathbb{E}_{q(\epsilon)}[\frac{\partial f(\boldsymbol{\omega}, \theta)}{\partial \boldsymbol{\omega}}\frac{\partial \boldsymbol{\omega}}{\partial \theta} + \frac{\partial f(\boldsymbol{\omega}, \theta)}{\partial \theta}]$$

- The deterministic function $t(\theta, \epsilon)$ transforms a sample of parameter-free noise (ϵ) and the variational posterior parameters θ into a sample from the variational posterior
- Let $f(\boldsymbol{\omega}, \theta) = logq_\theta(\boldsymbol{\omega}) - \log p(\boldsymbol{\omega}) - logp(D|\boldsymbol{\omega})$, we use the unbiased estimates of gradients of the cost $F(D, \theta)$, with $F(D, \theta)$ being approximated by $\sum_{i=1}^{n} \log q_\theta(\boldsymbol{\omega}^{(i)}) - logp(\boldsymbol{\omega}^{(i)}) - logp(D|\boldsymbol{\omega}^{(i)})$ using Monte Carlo sampling (note that each $\boldsymbol{\omega}^{(i)}$ is drawn from the variational posterior $q_\theta(\boldsymbol{\omega}^{(i)})$)

Bayes by Backprop

- Gaussian variational posterior :
  - Assume the variational posterior of weights follows the Gaussian distribution
  - To ensure the non-negativity of the standard deviation σ, parameterize it as $\sigma = \log(1 + \exp(\rho))$
  - Variational posterior parameters: $\theta = (\mu, \rho)$
  - $\boldsymbol{\omega} = t(\theta, \epsilon) = \mu + \log(1 + \exp(\rho)) \circ \epsilon$

Algorithm:

1. Sample $\epsilon \sim N(0, 1)$
2. Let $\boldsymbol{\omega} = \mu + \log(1 + \exp(\rho)) \circ \epsilon$
3. Let $\theta = (\mu, \rho)$
4. Let $f(\boldsymbol{\omega}, \theta) = log q_\theta(\boldsymbol{\omega}) - \log p(\boldsymbol{\omega}) - log p(D|\boldsymbol{\omega})$
5. Calculate the gradient with respect to the mean

$$\Delta_\mu = \frac{\partial f(\boldsymbol{\omega}, \theta)}{\partial \boldsymbol{\omega}} + \frac{\partial f(\boldsymbol{\omega}, \theta)}{\partial \mu}$$

6. Calculate the gradient with respect to the standard deviation parameter

$$\Delta_\rho = \frac{\partial f(\boldsymbol{\omega}, \theta)}{\partial \boldsymbol{\omega}} \frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial f(\boldsymbol{\omega}, \theta)}{\partial \rho}$$

7. Update the variational parameters:

$$\mu \leftarrow \mu - \alpha \, \Delta_\mu$$
$$\rho \leftarrow \rho - \alpha \, \Delta_\rho$$

Bayes by Backprop

- Spike-and-Slab prior :
  - $p(\boldsymbol{\omega}) = \prod_j \pi N(\omega_j|0,\sigma_1^2) + (1-\pi)N(\omega_j|0,\sigma_2^2)$ where ω_j is the jth weight of the neural network
  - Fix the hyper-parameter during training. Optimize the parameter in prior is not useful and yield worse results

- Minibatches and KL re-weighting:
  - Minibach: for each epoch of optimization, training data D is randomly split into a partition of M equally-sized subsets
  - Each gradient is averaged over all elements in one of these minibatches
  - KL cost can be partitioned non-uniformly among the minibatches at each epoch:

$$\pi \in [0,1]^M, \sum \pi_i = 1, \qquad F_i^{\pi}(D_i,\theta) = \pi_i KL\{q_\theta(\boldsymbol{\omega}) \parallel p(\boldsymbol{\omega})\} - \mathbb{E}_{q_\theta(\boldsymbol{\omega})}[logp(D_i|\boldsymbol{\omega})]$$

Then $\mathbb{E}_M[\sum F_i^{\pi}(D_i,\theta)] = F(D,\theta)$. Specifically, the scheme $\pi_i = \frac{2^{M-i}}{2^M-1}$ works well

- Advantages:
  - Introduce uncertainty on weights and thus we can estimate uncertainty on prediction
  - Flexibility: not use the closed form of the complexity cost (or entropic) part allows many more combinations of prior and variational posterior families
  - Easy to implement

- Disadvantages:
  - The number of parameters is doubled
  - The performance in prediction is not better than a closed-form KL divergence cost function

Uncertainty Quantification

- Variational predictive distribution:
  - We are interested in predictive distribution $p_D(y^*|x^*) = \int p_{\boldsymbol{\omega}}(y^*|x^*)p(\boldsymbol{\omega}|D)d\boldsymbol{\omega}$
  - We obtain the posterior samples of $\boldsymbol{\omega}$, thus we can also get the posterior predictive samples of y^*
  - We can get the posterior predictive mean and 95% credible intervals to measure the uncertainty

- Entropy based predictive uncertainty:
  - Gal (2016) introduced a Shannon-entropy-based predictive uncertainty for classification given by

$$H\{p(y^* \mid x^*, \mathcal{D})\} := -\sum_{k=1}^{K} p(y^* = e_k \mid x^*, \mathcal{D}) \log\{p(y^* = e_k \mid x^*, \mathcal{D})\}$$

  and suggested an easy-to-compute estimator given by $-\sum_{k=1}^{K} \hat{p}_{\hat{\theta}}(y^* = e_k \mid x^*) \log\{\hat{p}_{\hat{\theta}}(y^* = e_k \mid x^*)\}$.

- Aleatoric uncertainty and epistemic uncertainty:
  - Predictive variance can be decomposed into 2 parts: aleatoric and epistemic uncertainty

$$\mathrm{Var}_q\big(p(y^*|x^*)\big) = \mathbb{E}_q[yy^T] - \mathbb{E}_q[y]\mathbb{E}_q[y]^T = \underbrace{\frac{1}{T}\sum_{t=1}^{T} \mathrm{diag}(\hat{p}_t) - \hat{p}_t\ \hat{p}_t^T}_{\text{aleatoric}} + \underbrace{\frac{1}{T}\sum_{t=1}^{T}(\hat{p}_t - \bar{p})(\hat{p}_t - \bar{p})^T}_{\text{epistemic}}$$

  where $\bar{p} = \frac{1}{T}\sum_{t=1}^{T} \hat{p}_t$ and $\hat{p}_t = \mathrm{Softmax}\big(f_{w_t}(x^*)\big)$

  - Aleatoric uncertainty is a measure for the variation of data (data quality), epistemic uncertainty is caused by the model (model performance)
  - Popular in medical images

Case Study: Simpson Characters Recognition

- Image classification of 13 different Simpson characters under different methods, especially for non-Bayesian CNN(regularCNN) and BayesCNN
- To ensure our training data is balanced, the maximum number of images we use to train our model for each character is 1000 (for example, 1354 images have been found for Lisa Simpson, we only use the first 1000 of them).
- The training data contains 12773 images, and the testing set contains 641 images.
- Data pre-processing:
  - ***Cleaning***: remove the image containing multiple characters
  - ***Re-sizing***: load the image as size of 64*64
  - ***Normalization***: normalize the data by dividing 255 (reduce the effect of illumination)
  - ***One-hot Encoding***: encode labels to hot vectors as response variable for CNNs

Case Study: Simpson Characters Recognition

- The structure of the CNN:

Layer Type	Filters	Padding
Convolutional (3*3)	32	0
Convolutional (3*3)	32	0
Max-pooling (2*2)		
Convolutional (3*3)	64	0
Convolutional (3*3)	64	0
Max-pooling (2*2)		
Convolutional (3*3)	128	0
Convolutional (3*3)	128	0
Max-pooling (2*2)		
Fully-connected		

- Batch-size: 128
- Epochs: 20

- BayesCNN Model setting:

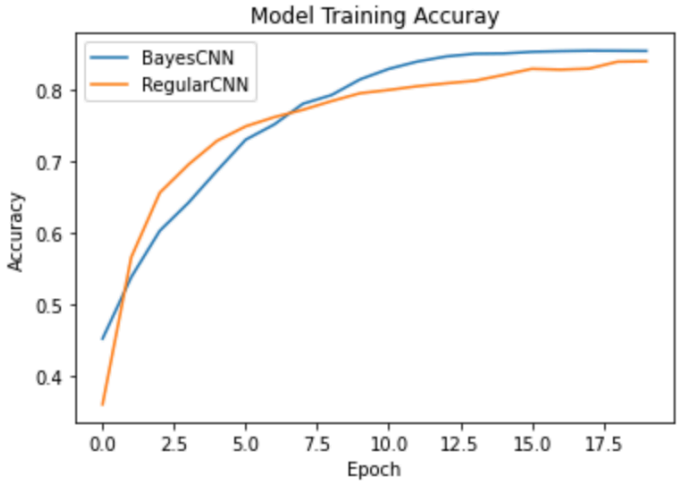
- ***Learning rate***: 10^{-3}
- $\boldsymbol{\pi}$: 0.25
- $\boldsymbol{\sigma_1^2}$: 1
- $\boldsymbol{\sigma_2^2}$: 0.005
- Initialization:
  - $\mu = 0$
  - $\rho \sim N(0, 10)$
- Same structure as CNN but with random weights in each CONV and Fully-connected layer

- The label of the characters:

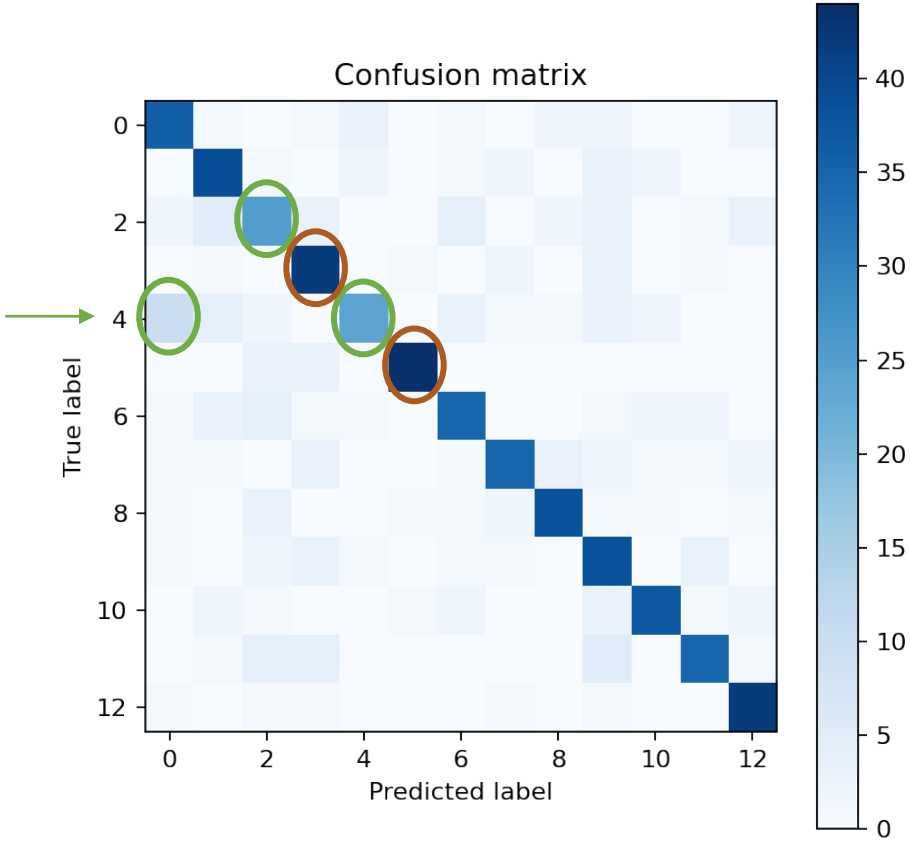
Label	Character Names
0	Abraham Grampa Simpson
1	Bart Simpson
2	Charles Montgomery Burns
3	Chief Wiggum
4	Homer Simpson
5	Krusty the Clown
6	Lisa Simpson
7	Marge Simpson
8	Milhouse Van Houten
9	Moe Szyslak
10	Ned Flanders
11	Principal Skinner
12	Sideshow Bob

Case Study: Simpson Characters Recognition

- Comparisons of two models in training and testing accuracy:



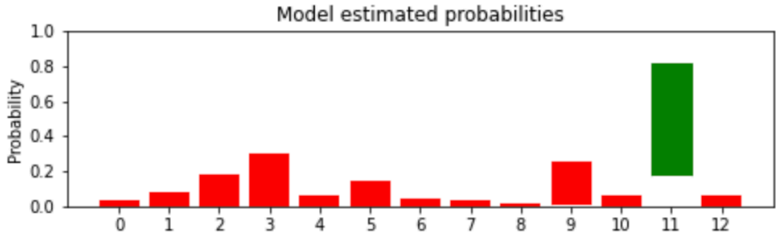
- Confusion matrix of RegularCNN:



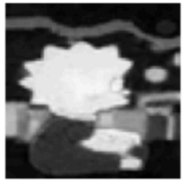
- Charles Montgomery Burns (label 2) and Homer Simpson (label 4) are more confusing with other characters. Specifically, Homer Simpson is easily mis-recognized as Grampa Simpson (label 0), which makes sense!
- Chief Wiggum (label 3) and Krusty the Clown (label 5) are easy to be recognized in the classifiers

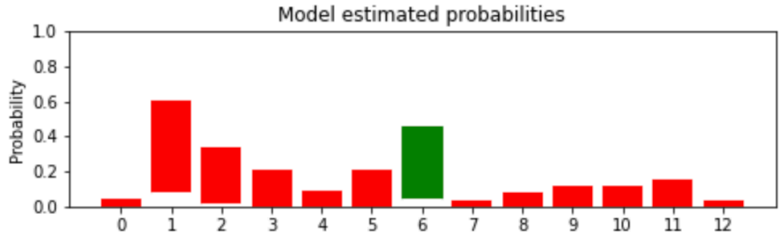
Case Study: Simpson Characters Recognition

True label: principal_skinner

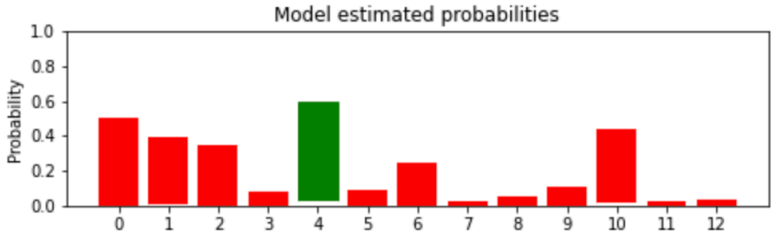


True label: lisa_simpson

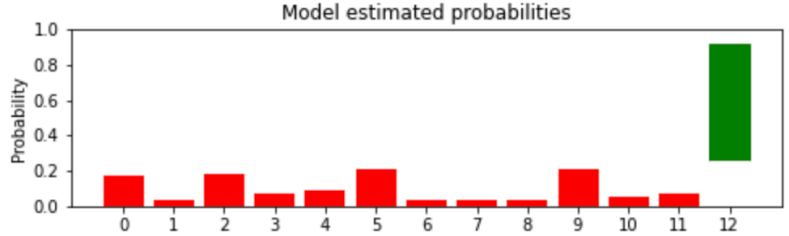




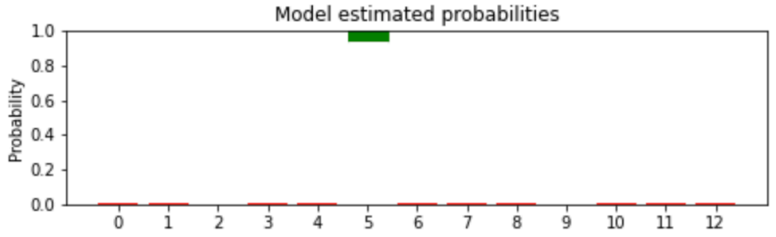
True label: homer_simpson



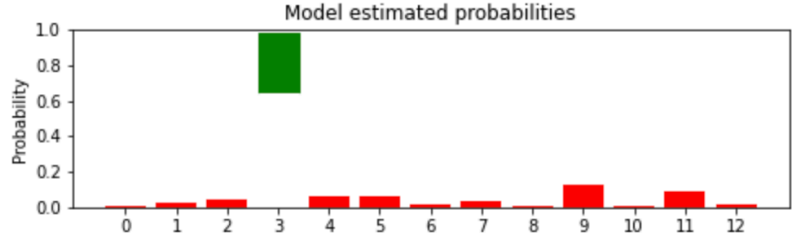
True label: sideshow_bob



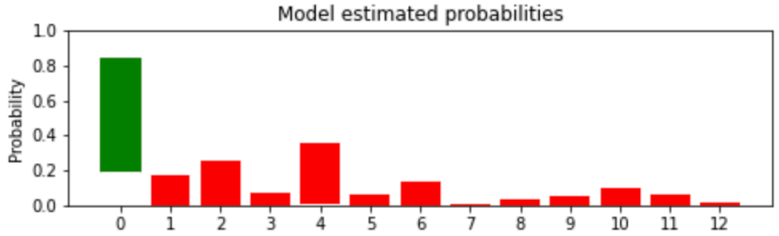
True label: krusty_the_clown



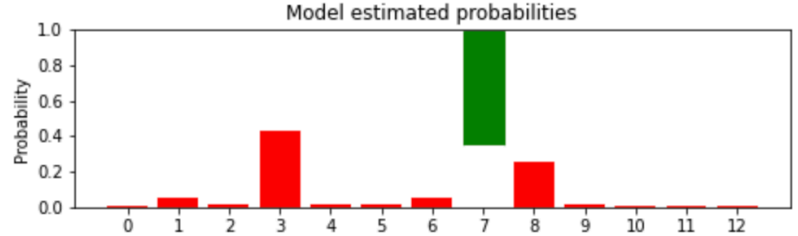
True label: chief_wiggum



True label: abraham_grampa_simpson



True label: marge_simpson



Thank you!