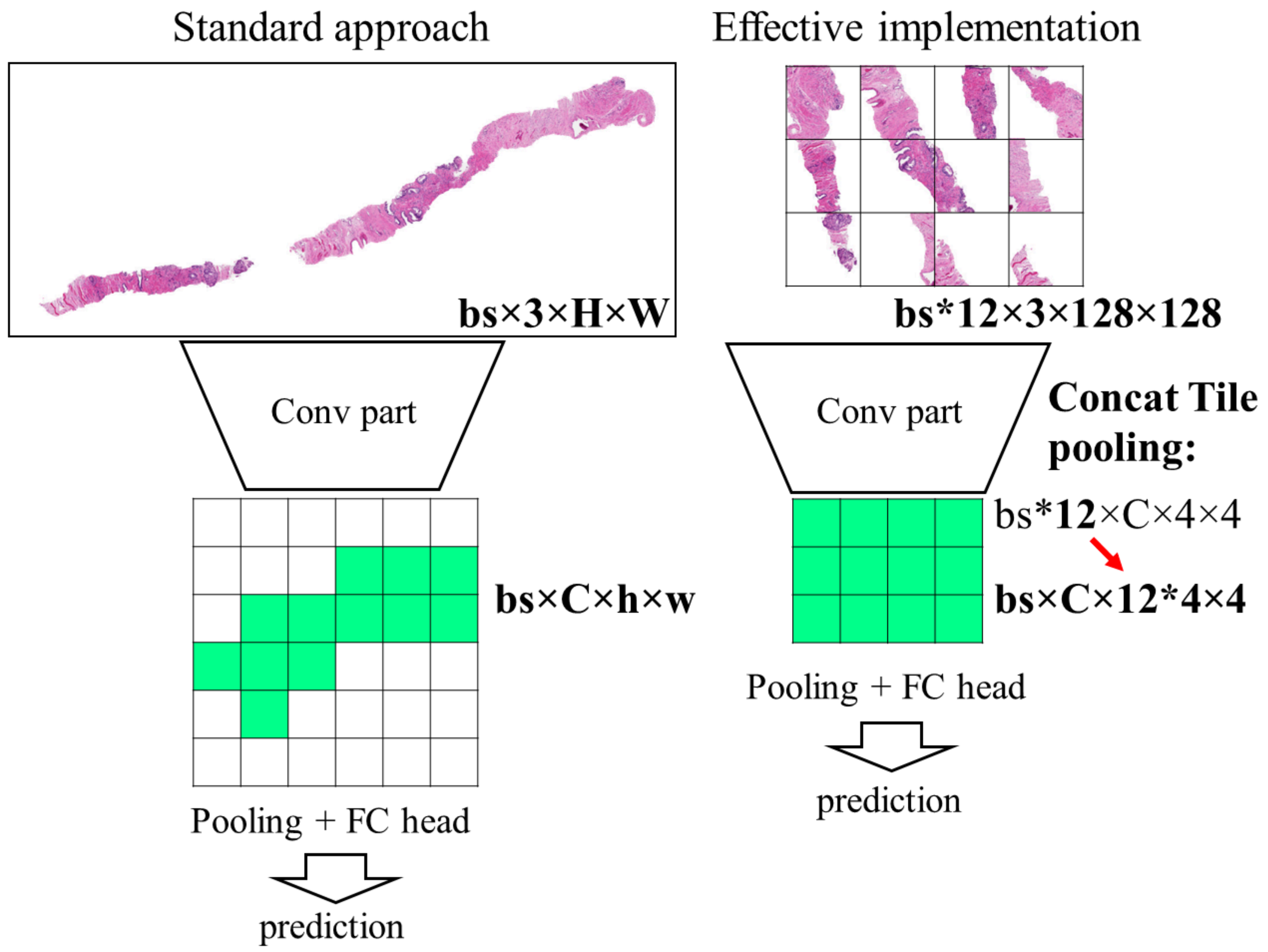
Kaggle Prostate Cancer Image Classification

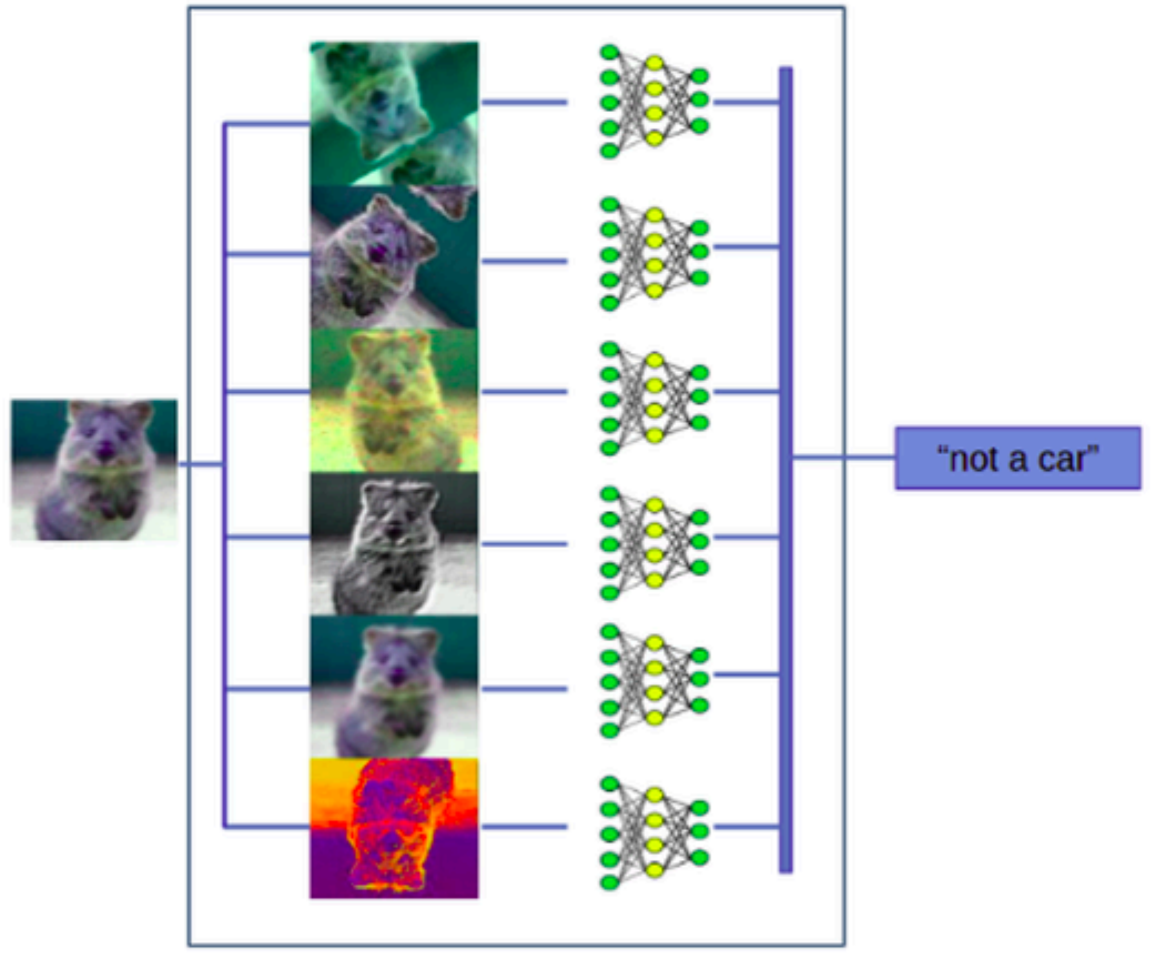
Kanru Wang

July 2020

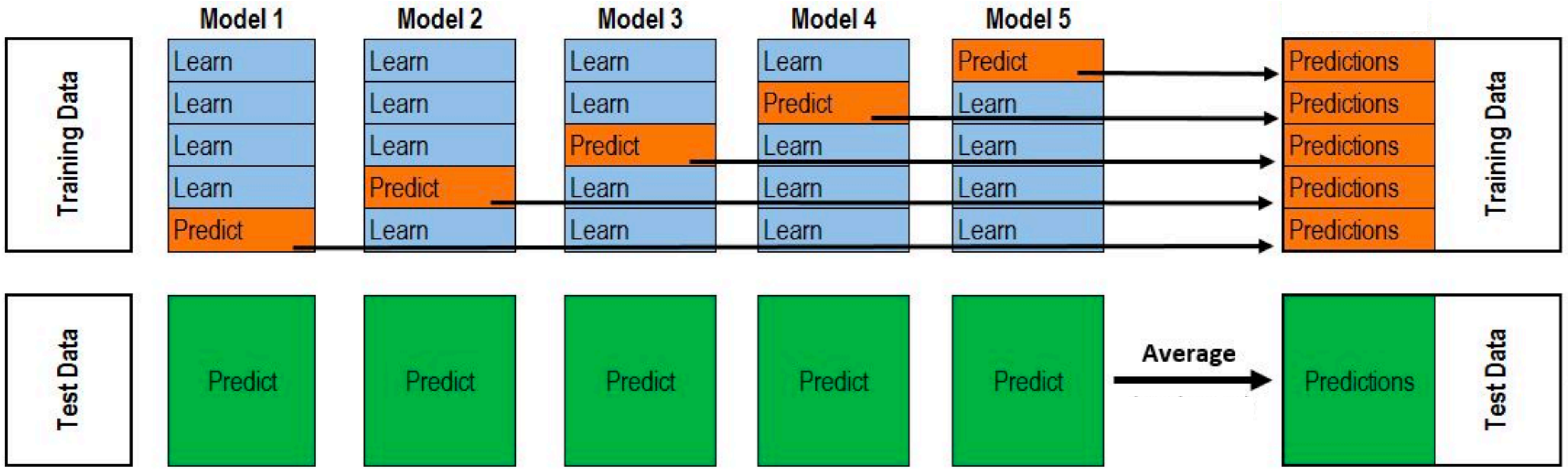
Normally we crop image to remove irrelevant surrounding space, but in this case we need tiling.
Notice that the 12 tiles are individually scored by the model, instead of merging into one image as below.



Test Time Augmentation (TTA)



Ensemble as an alternative to “train on entire dataset”



Assume 6 output classes, 4 ensemble models, 8 TTA. For each testing tile…

	Model 1								Model 2								Model 3								Model 4								
Class 1																																	Avg of 32
Class 2									8 TTA images																								Avg of 32
Class 3									1	2	3	4	5	6	7	8																	Avg of 32
Class 4																																	Avg of 32
Class 5																																	Avg of 32
Class 6																																	Avg of 32

The largest value will be the predicted class

For example: tile size = 128 x 128 x 3, number of tiles per image = 12, train batch size = 32, number of classes = 6, inference batch size = 2, number of TTA images = 8, number of ensemble models = 4

Initial data preparation

1. [original_height, original_width, 3]
2. Divide into tiles:
   [original_height // 128, 128, original_width // 128, 128, 3]
3. Pick top 12 tile with least blank space:
   [number of tiles (12), 128, 128, 3]

Training data preparation

1. Read in each tile with pil2tensor:
   [3, 128, 128]
2. Output of FastAI DataBunch is a list of 12 of such:
   [train batch size, 3, 128, 128]

Inference data preparation

1. Output of pytorch Dataset and then DataLoader
   [inference batch size, number of tiles, 3, 128, 128]
2. After TTA
   [inference batch size, number of TTA, number of tiles, 3, 128, 128]
3. Temporarily hide "number of TTA" in "batch size"
   [inference batch size x number of TTA, number of tiles, 3, 128, 128]

4 ensemble models, run 4 times

Model

1. Reshape to the expected shape of the pre-trained model
   [number of tiles x batch size, 3, 128, 128]
2. Output of the pre-trained model
   [number of tiles x batch size, constant 2048, constant 4, constant 4]
3. Reshape to the expected shape of the output layer
   [batch size, constant 2048, number of tiles x constant 4, constant 4]
4. Output of the output layer
   [batch size, number of classes]

Post-processing for inference

1. Stack the ensemble model output
   [inference batch size x number of TTA, number of ensemble models, number of classes]
2. Pool "number of TTA" and "number of ensemble models"
   [inference batch size, number of TTA x number of ensemble models, number of classes]
3. Calculate the mean over the pool
   [inference batch size, number of classes]
4. Argmax over the number of classes
   [inference batch size]

Cohen’s Kappa

Classification problems with ordered labels.
Treat it as a regression problem, and then post-process the predictions and convert them to integer ratings.

For example, the fitting process will find the best 4 thresholds (in terms of kappa score) for the 5 ordered classes.

```
[-2.5, 0.3, 0.4, 0.5, 1.2, 1.41, 1.5, 2.4, 2.42, 2.5, 3.4, 3.43, 3.5, 4, 9]
[0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4]
```

```
kappa_optimizer.fit(training_pred_prob, training_truth_label)
kappa_optimizer.predict(test_pred_prob)
```