Supplementary Materials:

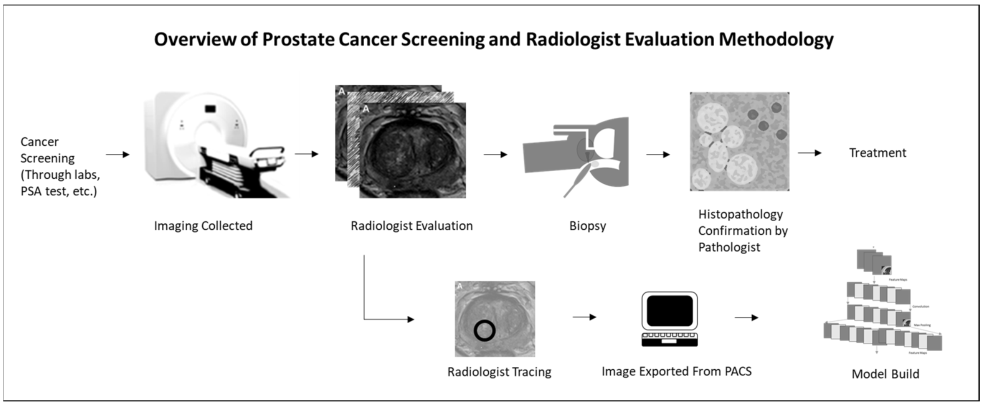

Figure S1. Overview of prostate cancer screening and Radiologist evaluation methodology on local dataset.

Table S1. Overview of total parameters and description of model layers for model developed on ProstateX using transfer learning on the ResNet model with Imagenet weights from the TensorFlow applications package. The ReLU activation function was used for each layer with a binary cross-entropy loss function. The "optimizers RMSprop" optimizer was used with a learning rate of 2e-5.

Model: "sequential_11"		
Layer (type)	Output Shape	Param #
functional_23 (Functional)	(None, 524288)	23587712
dense_73 (Dense)	(None, 512)	268435968
dropout_35 (Dropout)	(None, 512)	0
dense_74 (Dense)	(None, 512)	262656
dropout_36 (Dropout)	(None, 512)	0
dense_75 (Dense)	(None, 512)	262656
dropout_37 (Dropout)	(None, 512)	0
dense_76 (Dense)	(None, 512)	262656
dropout_38 (Dropout)	(None, 512)	0
dense_77 (Dense)	(None, 1)	513
Total params: 292,812,161		
Trainable params: 269,224,449		
Non-trainable params: 23,587, 712		

Table S2. Overview of code used to create algorithms for each training set.

```
#Package Imports
import tensorflow as tf
from tensorflow import keras
from keras.applications.resnet50 import ResNet50
from tensorflow.keras import models
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, InputLayer
from tensorflow.keras.models import Sequential
from tensorflow.keras import optimizers
```

```
#Parameters
batch_size = 32
IMG_ HEIGHT = 512
IMG_WIDTH = 512

#training dataset = aug_ds and validation set = val_ds
```

```
#Prepare the model, for this we're using a pretrained model on "imagenet" data
modnet = modeltype
output = modnet.layers[-1].output
output2 = keras.layers.Flatten()(output)
modnet = keras.Model(modnet.input, outputs=output2)

#Set layers to not trainable to just use the last layer for transfer learning and provide a preliminary summary
for layer in modnet.layers:
    layer.trainable = False
modnet.summary()
```

```
model = Sequential()
model.add(modnet)

model.add(Dense(512, activation='relu', input_dim=(IMG_HEIGHT,IMG_WIDTH,3)))
model.add(Dropout(0.3))

model.add(Dense(512, activation='relu', input_dim=(IMG_HEIGHT,IMG_WIDTH,3)))
model.add(Dropout(0.3))

model.add(Dense(512, activation='relu', input_dim=(IMG_HEIGHT,IMG_WIDTH,3)))
model.add(Dropout(0.3))

model.add(Dense(512, activation='relu', input_dim=(IMG_HEIGHT,IMG_WIDTH,3)))
model.add(Dropout(0.3))

model.compile(loss='binary_crossentropy',
        optimizer=optimizers.RMSprop(lr=2e-5),
        metrics=['accuracy', 'mean_absolute_error', 'mean_squared_error'])
model.summary()
```

```
history = model.fit(aug_ds.repeat(),
            steps_per_epoch=int(100/batch_size),
            epochs=epocsnum,
            validation_data=val_ds.repeat(),
            validation_steps=int(50/batch_size),
            callbacks=[csv_logger, time_callback])
```

Table S3. Overview of equations used within ResNet model and validation tests.

Overview	Equation
ReLu Activation Function	f(x) = max(0,X)
RMSProp Optimizer Function	$w_{t+1} = w_t + (-(L_r / sqrt(ExpAvgG)*G)$ Where G is the Gradient at time t along w
t-test	t = (m – v) / (S/ sqrt(n)) Where m = mean, v = value, S = StDev, and n = Sample size

Table S4. Table 3 including Standard Deviations.

ResNet Model	Training image source	Testing image source	Training & testing image sequence					
			T2		ADC		T2 & ADC	
			AUC	Accuracy	AUC	Accuracy	AUC	Accuracy
Model^{PX2}	PX2 (89)	PX2*	0.93 ± 0.01	0.91 ± 0.03	0.91 + 0.01	0.88 + 0.01	0.95 + 0.01	0.90 + 0.01
		Local†	0.49 ± 0.01	0.53 ± 0.02	0.23 ± 0.00	0.48 ± 0.00	0.46 ± 0.04	0.55 ± 0.03
		PXL†	0.87 ± 0.01	0.79 ± 0.01	0.80 ± 0.02	0.78 ± 0.02	0.78 ± 0.03	0.80 ± 0.02
Model^{Loc}	Local	Local*	0.96 ± 0.01	0.89 ± 0.02	0.82 ± 0.01	0.82 ± 0.00	0.98 ± 0.01	0.92 ± 0.02
		PX2†	0.50 ± 0.01	0.54 ± 0.01	0.49 ± 0.01	0.49 ± 0.00	0.41 ± 0.02	0.51 ± 0.01
		PXL†	0.77 ± 0.01	0.71 ± 0.01	0.84 ± 0.01	0.84 ± 0.00	0.94 ± 0.3	0.87 ± 0.03
Model^{PXL}	PXL	PXL*	0.83 ± 0.01	0.89 ± 0.01	0.98 ± 0.02	0.92 ± 0.01	0.96 ± 0.01	0.93 ± 0.02
		PX2†	0.92 ± 0.02	0.92 ± 0.02	0.85 ± 0.01	0.91 ± 0.00	0.85 ± 0.01	0.93 ± 0.04
		Local†	0.96 ± 0.01	0.86 ± 0.03	0.88 ± 0.02	0.92 ± 0.01	0.99 ± 0.05	0.92 ± 0.05

Table S5. Standard Deviations for Table 4.

ResNet Algorithm	Training image source	Testing image source	Training & testing image sequence											
			T2				ADC				T2 & ADC			
			AUC		Accuracy		AUC		Accuracy		AUC		Accuracy	
			PZ	TZ	PZ	TZ	PZ	TZ	PZ	TZ	PZ	TZ	PZ	TZ
Model^{PX2}	PX2	PX2*	0.92 ± 0.01	0.93 ± 0.01	0.91 ± 0.00	0.90 ± 0.03	0.91 ± 0.01	0.91 ± 0.01	0.88 ± 0.01	0.88 ± 0.01	0.94 ± 0.01	0.95 ± 0.01	0.91 ± 0.01	0.90 ± 0.01
		Local†	0.44 ± 0.01	0.61 ± 0.00	0.53 ± 0.01	0.55 ± 0.02	0.23 ± 0.01	0.25 ± 0.02	0.48 ± 0.00	0.48 ± 0.01	0.45 ± 0.01	0.31 ± 0.02	0.44 ± 0.01	0.44 ± 0.00
		PXL†	0.88 ± 0.01	0.87 ± 0.03	0.8 ± 0.01	0.77 ± 0.01	0.80 ± 0.01	0.81 ± 0.01	0.77 ± 0.02	0.80 ± 0.01	0.77 ± 0.01	0.79 ± 0.02	0.80 ± 0.01	0.80 ± 0.02
Model^{Loc}	Local	Local*	0.88 ± 0.01	0.99 ± 0.03	0.86 ± 0.01	0.89 ± 0.01	0.82 ± 0.01	0.81 ± 0.02	0.84 ± 0.01	0.80 ± 0.03	0.95 ± 0.01	0.96 ± 0.03	0.90 ± 0.02	0.94 ± 0.04
		PX2†	0.48 ± 0.02	0.52 ± 0.01	0.53 ± 0.02	0.55 ± 0.01	0.54 ± 0.02	0.45 ± 0.01	0.49 ± 0.02	0.48 ± 0.02	0.55 ± 0.01	0.29 ± 0.02	0.57 ± 0.01	0.46 ± 0.02
		PXL†	0.79 ± 0.02	0.69 ± 0.02	0.79 ± 0.01	0.70 ± 0.03	0.94 ± 0.02	0.72 ± 0.01	0.84 ± 0.01	0.80 ± 0.02	0.89 ± 0.04	0.94 ± 0.02	0.83 ± 0.02	0.88 + 0.02
Model^{PXL}	PXL	PXL*	0.83 ± 0.01	0.92 ± 0.02	0.89 ± 0.02	0.90 ± 0.02	0.98 ± 0.01	0.99 ± 0.02	0.92 ± 0.01	0.93 ± 0.01	0.96 ± 0.01	0.90 ± 0.01	0.91 ± 0.01	0.98 ± 0.03
		PX2†	0.93 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.88	0.85	0.92	0.90	0.85	0.86	0.93	0.92
		Local†	0.96 ± 0.01	0.95 ± 0.02	0.94 ± 0.03	0.78 ± 0.01	0.90	0.88	0.92	0.91	0.99	0.97	0.92	0.92