

Image Classification on an FPGA

Arsal Zaman, Horace Lee, Kimon Grigorakis, Moin Bukhari























Can a neural network learn to recognize doodling?

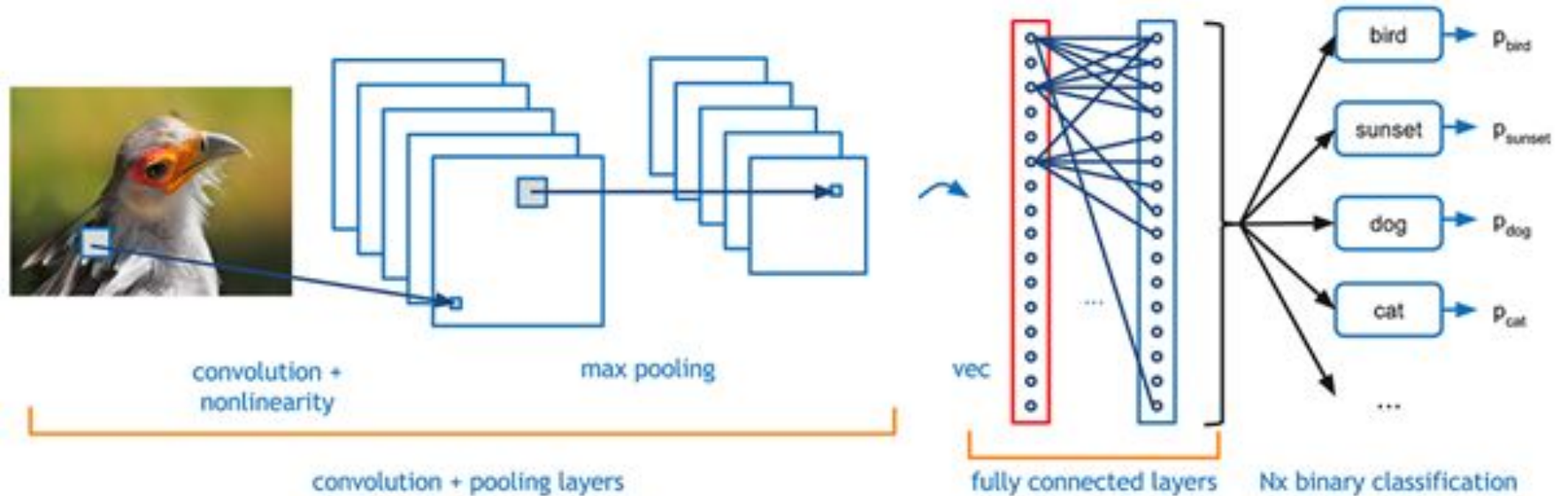
Help teach it by adding your drawings to the [world's largest doodling data set](#), shared publicly to help with machine learning research.

Let's Draw!

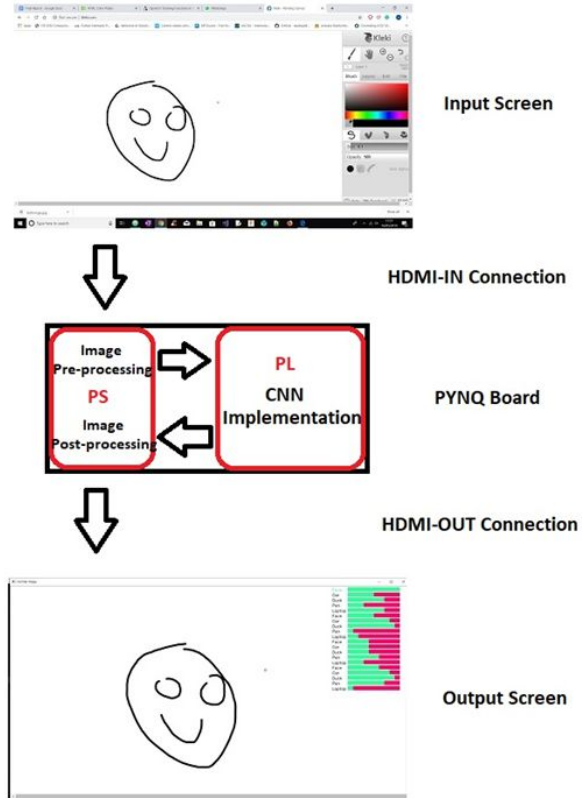
Classes used (with example images)

airplane 	apple 	bathtub 	bicycle 	bird 	brain 	car 
cat 	cell phone 	dolphin 	flower 	peas 	penguin 	shoe 
skyscraper 	speedboat 	television 	violin 	watermelon 	wristwatch 	

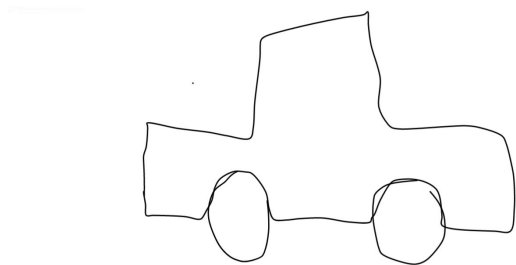
Convolutional Neural Network



High Level Description



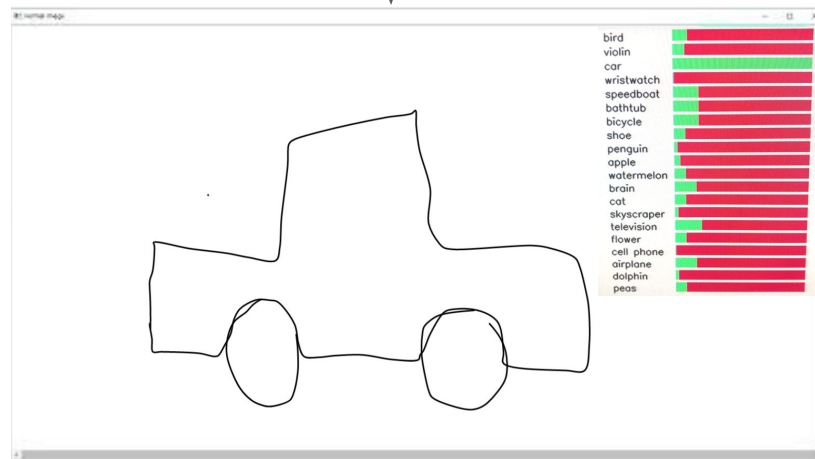
Process



Input Frame



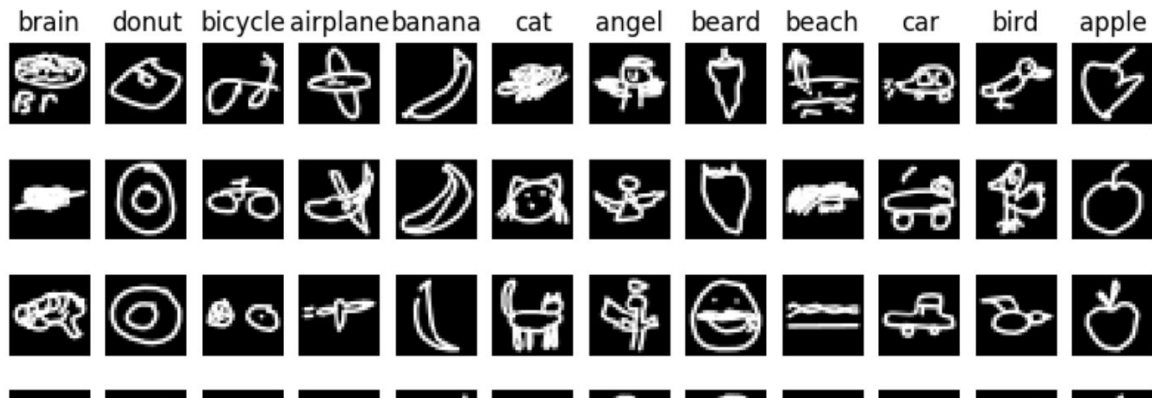
Adjusted Frame
for CNN

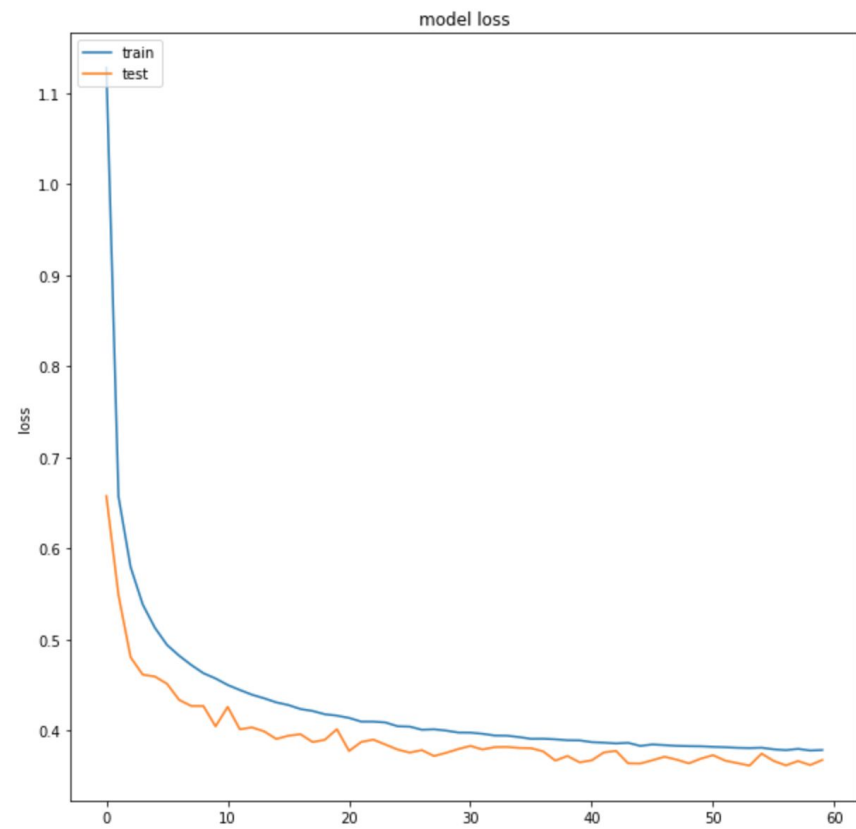


Output
Frame

Kiddo

Playing with Google Quick draw dataset with Keras. Dataset can be found [here](#) Kiddo is a convolution neural network model designed to identify hand drawn images. The speciality of these images is that these are drawn with a very small interval of time. The images in a single class are not drawn by all different person in small time, so they are least possible to have any sort of similarities. 10 images from the 10 datasets we used are shown below:





Decreased parameter count and MACs -> runs faster, uses less resources

Accuracy ~93% -> ~89%

Reduced number of kernels, reduced number of neurons in fully connected layers

Added 1x1 convolution (i.e. pointwise convolution) and maxpooling

Original architecture			
Layer	Output dimensions (H, W, C)	MACs	Parameter count
Input	28, 28, 1	-	-
3x3 conv with padding	28, 28, 6	42,336	60
3x3 conv	26, 26, 32	1,168,128	1760
2x2 maxpool	13, 13, 32	-	-
dropout	13, 13, 32	-	-
3x3 conv with padding	13, 13, 64	3,115,008	18496
3x3 conv	11, 11, 64	4,460,544	36928
2x2 maxpool	5, 5, 64	-	-
dropout	5, 5, 64	-	-
flatten	1600	-	-
fully connected	512	819,200	819712
dropout	512	-	-
fully connected (softmax activation)	20	10,240	10260
Total		9,615,456	887,216

Our current architecture			
Layer	Output dimensions (H, W, C)	MACs	Parameter count
Input	28, 28, 1	-	-
3x3 conv with padding	28, 28, 6	42,336	60
3x3 conv	26, 26, 24	876,096	1320
1x1 conv (pointwise conv)	26, 26, 12	194,688	300
2x2 maxpool	13, 13, 12	-	-
dropout	13, 13, 12	-	-
3x3 conv	13, 13, 28	511,056	3052
2x2 maxpool	6, 6, 28	-	-
1x1 conv (pointwise conv)	6, 6, 24	24,192	696
2x2 maxpool	3, 3, 24	-	-
dropout	3, 3, 24	-	-
flatten	216	-	-
fully connected	100	21,600	21700
dropout	100	-	-
fully connected	20	2,000	2020
Total		1,673,968	29,148

Quantization / custom precision

```
#include "hls_half.h"
```

```
typedef half quant_t;
```

```
const quant_t conv2D_1_kernels[6][3][3][1] = {{{{ 0.22381108 },
  {-0.29261965 },
  {-0.2661797  }},
  {{{-0.087646745},
  { 1.2625207  },
  { 0.7322538  }},
  {{{-0.29847005 },
  {-0.9013102  },
  {-0.39091668 }}}},
  {{{ 0.028393432},
  { 0.029059542},
  {-0.04945678  }},
  {{{-0.24063097 },
  {-0.40525627 },
  { 0.13201526  }},
  {{{-0.7233055  },
  { 0.45510665  },
  { 0.88700783  }}}},
```

~2x decrease in
FF and LUT
utilization

Pipelining

```
// conv2D_3
```

```
H = 26; W = 26; C = 24; N_k = 12; D_k = 1;
```

```
H_output = 26; W_output = 26;
```

```
quant_t conv2D_3[26][26][12];
```

```
for (uint8_t n_k = 0; n_k < N_k; n_k++) {
```

```
    for (uint8_t h_output = 0; h_output < H_output; h_output++) {
```

```
        for (uint8_t w_output = 0; w_output < W_output; w_output++) {
```

```
#pragma HLS PIPELINE
```

```
            quant_t acc = 0;
```

```
            for (uint8_t i = 0; i < D_k; i++) {
```

```
                for (uint8_t j = 0; j < D_k; j++) {
```

```
                    for (uint8_t c = 0; c < C; c++) {
```

```
                        acc += conv2D_2[h_output + i][w_output + j][c] * conv2D_3_kernels[n_k][i][j]
```

```
                    }
```

```
                }
```

```
            }
```

```
            acc += conv2D_3_biases[n_k];
```

```
            if (acc < 0) conv2D_3[h_output][w_output][n_k] = 0;
```

```
            else conv2D_3[h_output][w_output][n_k] = acc;
```

```
        }
```

```
    }
```

```
}
```

Working performance

CNN overall latency (not including other overheads):

13.98 ms (71fps)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	7.360	5.00

- [-] **Latency (clock cycles)**

Summary

Latency		Interval		Type
min	max	min	max	
1899381	1899381	1899381	1899381	none

Detail

Instance

⊕ Loop

Utilization Estimates

Summary

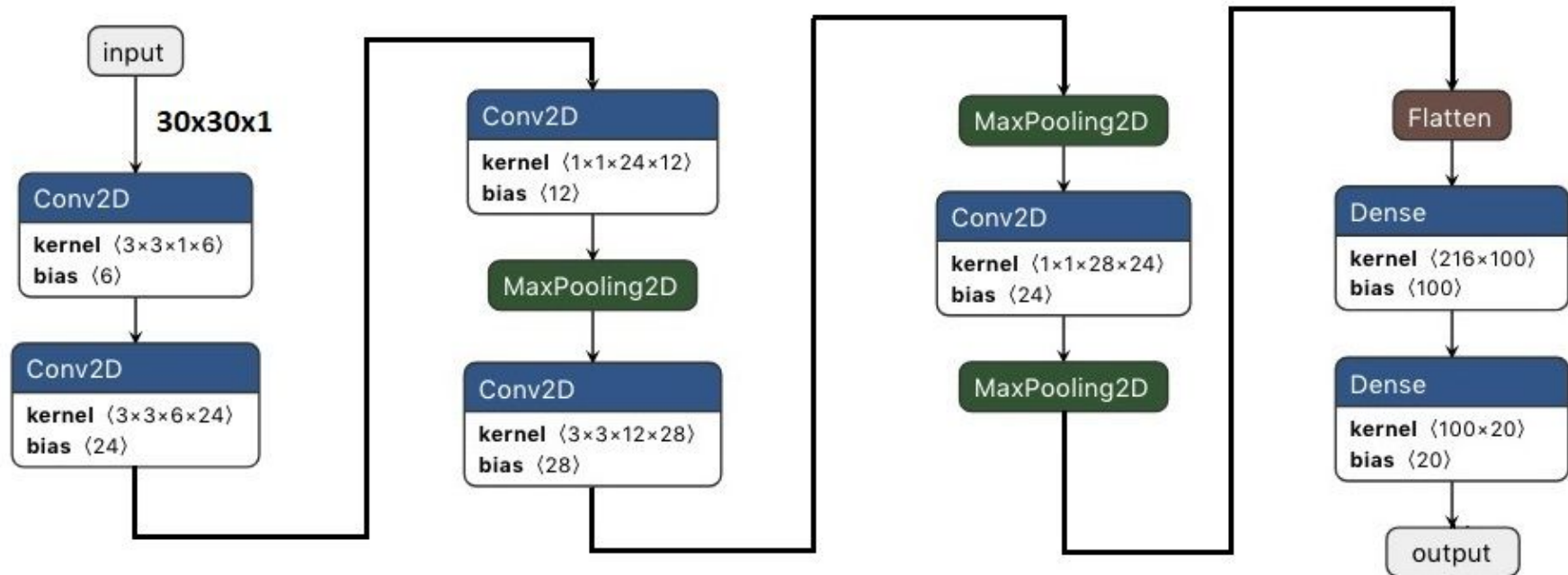
Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	3	-	-
Expression	-	0	0	9073
FIFO	-	-	-	-
Instance	2	10	1799	1387
Memory	92	-	1520	448
Multiplexer	-	-	-	5679
Register	0	-	16102	3840
Total	94	13	19421	20427
Available	280	220	106400	53200
Utilization (%)	33	5	18	38

INFO
INFO
INFO
Fin

```

/usr
INFO
INFO
INFO
INFO
INFO
INFO
INFO
INFO
INFO
INFO

```



Limitations

- Requires online sketchpad rather than real-world image
- Fixed input image size
- Specific to PYNQ-Z1 board