**End-to-End Demonstration of an Approximate Computing System**

**ACCEPT: C/C++ Compiler for Approximate Programs**

```c
ACCEPT float a;
float p;

a = p;  // approx add
p = a;  // approx add

A quality autotuner helps programmers navigate the quality/performance tradeoff space
```

**SNAP: Hardware Support for Neural Acceleration**

We leverage off-the-shelf programmable SoCs to implement an efficient and reconfigurable Neural Processing Unit.

**Evaluation Results**

- **Average application speedup**
  - Baremetal: $3.8 \times$, range: $1.3x - 38x$
  - Linux: $3.6 \times$, range: $1.1x - 7.7x$

- **Average energy reduction**
  - Baremetal: $2.8 \times$, range: $0.87x - 28x$
  - Linux: $2.8 \times$, range: $0.86x - 5.65x$

**Programming Model**

The programmer marks data that is approximable using type annotations. In addition, the programmer provides a quality metric and a set of representative test inputs.

**Example:** Image “Cartoonization”

```c
APPROX float cartoonize(float w[3][3]) {
    bool mask = diff of gauss(w) < -3e-3;
    return bilateral[w] * mask;
}
```

**Hybrid Binary**

Dynamic NPU configuration, neural network offloading

**Neural Acceleration**

- Approximates compute-intensive functions with neural networks which can then be efficiently evaluated on a hardware accelerator.

**Hardware Design**

- Tight coupling allows low-latency, coherent communication with the core.
- Processing Elements arranged in a systolic array make efficient use of DSPs.
- Replicated structure can take advantage of parallelism in the application.

**Tech Transfer**

This work was partly funded by Qualcomm through their Innovation Fellowship program.

**ACCEPT is an open-sourced** compiler accessible at [http://accept.rocks](http://accept.rocks)

**Publications & Reports**

- Sampson et al., ACCEPT: A Programmer-Guided Framework for Practical Approximate Computing, UW-TR
- Moreau et al., SNAPP: Approximate Computing on Programmable SoCs via Neural Acceleration, HPCA2015