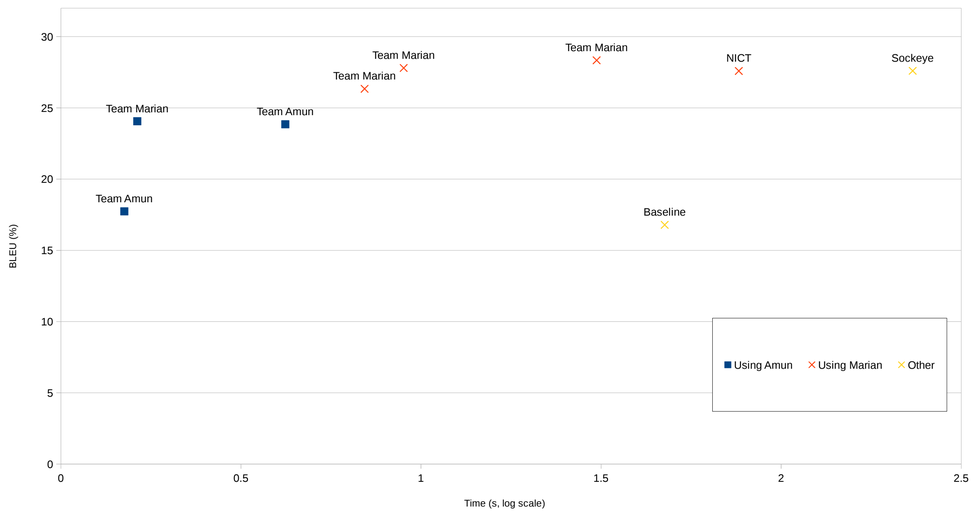
Fast Neural Machine Translation Implementation

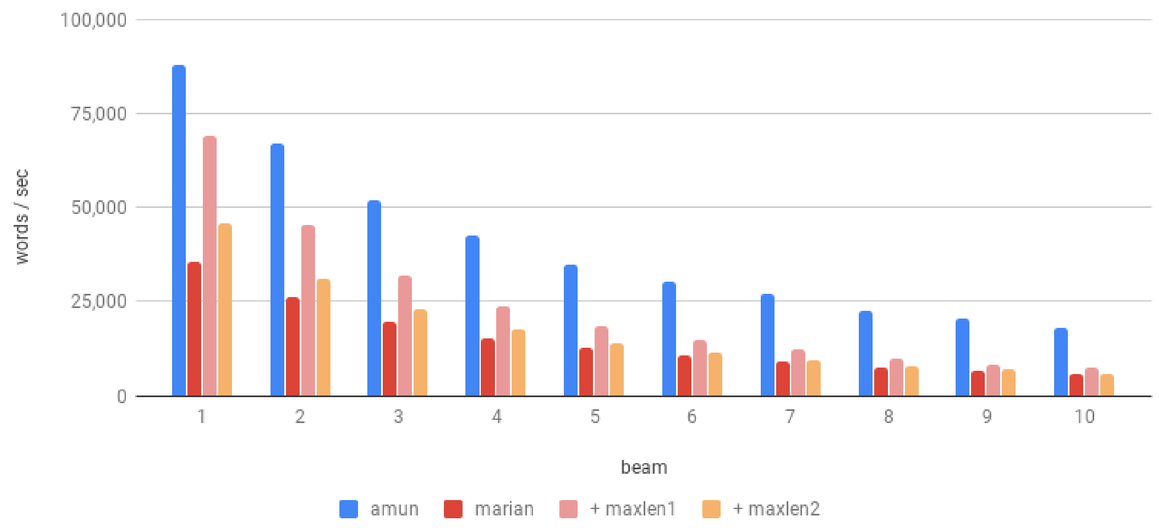
Hieu Hoang, Tomasz Dwojak, Rihards Krislauks
Daniel Torregrosa, Kenneth Heafield

Fastest translation engine for GPU

Official results of GPU track

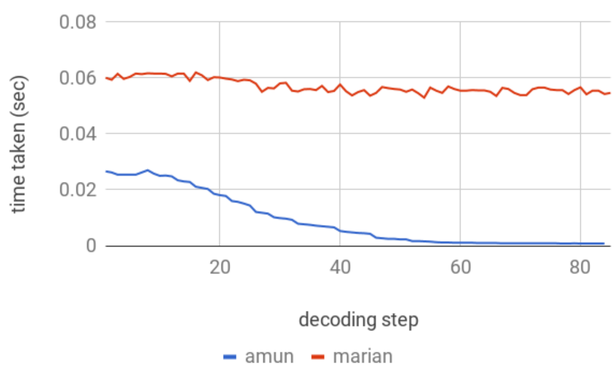


30% to 3+ times faster than Marian with the same model



Efficient Mini-Batching

Time taken in each decoding step (same batch):



Speed v Batch Size

Baseline

```
procedure BATCHING(encoded sentences i)
    Create batch b from i
    while hypo h ≠ EOS, ∀h ∈ b do
        Decode(b)
    end while
end procedure
```

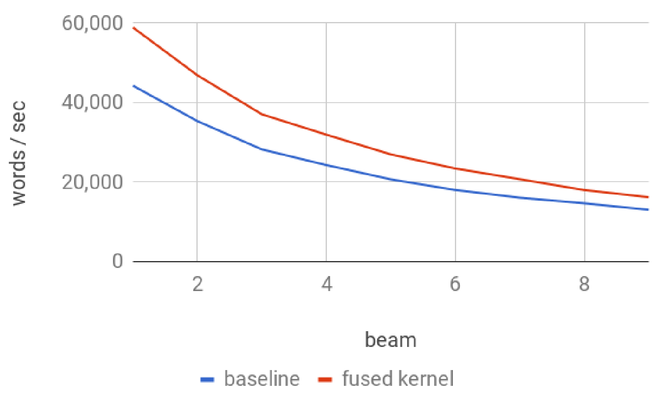
Our work

```
procedure BATCHING(encoded sentences i)
    Create batch b from i
    while b ≠ ∅ do
        Decode(b)
        for all hypo h ∈ b do
            if h = EOS then
                Remove h from b
            end if
        end for
    end while
end procedure
```

"Remove completed hypotheses from decoding"

Softmax / k-Best Fusion

Speed v. Beam size



Time taken:

	Baseline	Fused
Beam size 1		
Multiplication	5.39	5.38
Add bias	1.26	
Softmax	1.69	2.07 (-86.6%)
K-Best	12.53	
Beam size 3		
Multiplication	14.18	14.16
Add bias	3.76	
Softmax	4.75	3.43 (-87.1%)
K-Best	18.23	
Beam size 9		
Multiplication	38.35	38.42
Add bias	11.64	
Softmax	14.40	17.5 (-72.1%)
K-Best	36.70	

Baseline

```
procedure ADDBIAS(vector p, bias vector b)
    for all p_i in p do
        p_i ← p_i + b_i
    end for
end procedure

procedure SOFTMAX(vector p)
                ▷ calculate max for softmax stability
    max ← −∞
    for all p_i in p do
        if p_i > max then
            max ← p_i
        end if
    end for
                                ▷ calculate denominator
    sum ← 0
    for all p_i in p do
        sum ← sum + exp(p_i − max)
    end for
                                        ▷ calculate softmax
    for all p_i in p do
        p_i ← exp(p_i − max) / sum
    end for
end procedure

procedure FIND-BEST(vector p)
    max ← −∞
    for all p_i in p do
        if p_i > max then
            max ← p_i
            best ← i
        end if
    end for
     return max, best
end procedure
```

Our work

```
procedure FUSED-KERNEL(vector p, bias vector b)
    max ← −∞
    sum ← 0
    for all p_i in p do
        p'_i ← p_i + b_i
        if p'_i > max then
            Δ ← max − p'_i
            sum ← Δ × sum + 1
            max ← p'_i
            best ← i
        else
            sum ← sum + exp(p'_i − max)
        end if
    end for
     return 1/sum, best
end procedure
```

"Reduce the number of scans over activation matrix"

Tensor Cores

Hardware-accelerated matrix multiplication