UNIVERSITY OF PADUA
UNIVERSITA' DEGLI STUDI DI PADOVA

Zeroth-Order Frank-Wolfe Optimization

for Black-Box Adversarial Attacks

Optimization for Data Science, A.Y. 2023/24

Marco Uderzo, 2096998

DIPARTIMENTO
MATEMATICA

Introduction

- **Goal** of the project: comparing **Zeroth-Order** Frank-Wolfe variants in constraint optimization problems and testing on **black-box adversarial attacks** against MNIST.
- Implemented Algorithms:
  - **FZCGS**: Faster Zeroth-Order Conditional Gradient Sliding Method (Gao et al.)
  - **SGFFW**: Stochastic Gradient-Free Frank-Wolfe, with three different gradient approximation schemes:
    - **KWSA**: Kiefer-Wolfowitz Stochastic Approximation
    - **RDSA**: Random Directions Stochastic Approximation
    - **I-RDSA**: Improvised Random Directions Stochastic Approximation
- **Adversarial Attack**: deliberate **manipulation** of the **input** data with the intention of causing a **ML model** to make a mistake or produce **incorrect output**. This is done through **perturbation** of the input in a way that is not easily noticeable to a human observer.

FZCGS: Faster Zeroth-Order Conditional Gradient Sliding

Employs Coordinate-wise Gradient Estimator

$$\hat{\nabla} f(\mathbf{x}) = \sum_{j=1}^{d} \frac{f(\mathbf{x} + \mu_j \mathbf{e}_j) - f(\mathbf{x} - \mu_j \mathbf{e}_j)}{2\mu_j} \mathbf{e}_j$$

- $\hat{\nabla} f(\mathbf{x})$ is the estimated gradient of the function f in $\mathbf{x}$
- d is the dimensionality of the optimization space
- $\mu_j > 0$ is a smoothing parameter
- $\mathbf{e}_j \in \mathbb{R}^d$ is the basis vector where only the j-th element is 1 and all others are 0.

Algorithm 2 Faster Zeroth-Order Conditional Gradient Method (FZCGS)

Input: $\mathbf{x}_0, q > 0, \mu > 0, K > 0, \eta > 0, \gamma > 0, n$

for $k = 0, \cdots, K-1$ **do**
  if mod(k, q) = 0 **then**
    Sample S_1 without replacement to compute $\hat{\mathbf{v}}_k = \hat{\nabla} f_{S_1}(\mathbf{x}_k)$
  else
    Sample S_2 with replacement to compute $\hat{\mathbf{v}}_k = \frac{1}{|S_2|} \sum_{i \in S_2} [\hat{\nabla} f_i(\mathbf{x}_k) - \hat{\nabla} f_i(\mathbf{x}_{k-1}) + \hat{\mathbf{v}}_{k-1}]$
  end if
  $\mathbf{x}_{k+1} = \text{condg}(\hat{\mathbf{v}}_k, \mathbf{x}_k, \gamma_k, \eta_k)$
end for

Output: Randomly choose $\mathbf{x}_\alpha$ from $\{\mathbf{x}_k\}$ and return it

Algorithm 3 $\mathbf{u}^+ = \text{condg}(\mathbf{g}, \mathbf{u}, \gamma, \eta)$ (Qu et al., 2017)

1: $\mathbf{u}_1 = \mathbf{u}, t = 1$
2: $\mathbf{v}_t$ be an optimal solution for

$$V_{\mathbf{g},\mathbf{u},\gamma}(\mathbf{u}_t) = \max_{\mathbf{x} \in \Omega} \langle \mathbf{g} + \frac{1}{\gamma}(\mathbf{u}_t - \mathbf{u}), \mathbf{u}_t - \mathbf{x} \rangle$$

3: If $V_{\mathbf{g},\mathbf{u},\gamma}(\mathbf{u}_t) \leq \eta$, return $\mathbf{u}^+ = \mathbf{u}_t$.
4: Set $\mathbf{u}_{t+1} = (1 - \alpha_t)\mathbf{u}_t + \alpha_t \mathbf{v}_t$ where $\alpha_t = \min\{1, \frac{\langle \frac{1}{\gamma}(\mathbf{u} - \mathbf{u}_t) - \mathbf{g}, \mathbf{v}_t - \mathbf{u}_t \rangle}{\frac{1}{\gamma} \|\mathbf{v}_t - \mathbf{u}_t\|^2}\}$.
5: Set $t \leftarrow t + 1$ and goto step 2.

FZCGS: Faster Zeroth-Order Conditional Gradient Sliding

- Employs:
  - Coordinate-wise Gradient Estimation
  - Variance Reduction technique
  - Conditional Gradient Sliding

Algorithm 2 Faster Zeroth-Order Conditional Gradient Method (FZCGS)

Input: $\mathbf{x}_0, q > 0, \mu > 0, K > 0, \eta > 0, \gamma > 0, n$

1: **for** $k = 0, \cdots, K-1$ **do**
2: **if** mod(k, q) = 0 **then**
3: Sample S_1 without replacement to compute $\hat{\mathbf{v}}_k = \hat{\nabla} f_{S_1}(\mathbf{x}_k)$
4: **else**
5: Sample S_2 with replacement to compute $\hat{\mathbf{v}}_k = \frac{1}{|S_2|}\sum_{i \in S_2}[\hat{\nabla} f_i(\mathbf{x}_k) - \hat{\nabla} f_i(\mathbf{x}_{k-1}) + \hat{\mathbf{v}}_{k-1}]$
6: **end if**
7: $\mathbf{x}_{k+1} = \text{condg}(\hat{\mathbf{v}}_k, \mathbf{x}_k, \gamma_k, \eta_k)$
8: **end for**

Output: Randomly choose $\mathbf{x}_\alpha$ from $\{\mathbf{x}_k\}$ and return it

Algorithm 3 $\mathbf{u}^+ = \text{condg}(\mathbf{g}, \mathbf{u}, \gamma, \eta)$ (Qu et al., 2017)

1: $\mathbf{u}_1 = \mathbf{u}, t = 1$
2: $\mathbf{v}_t$ be an optimal solution for

$$V_{\mathbf{g},\mathbf{u},\gamma}(\mathbf{u}_t) = \max_{\mathbf{x} \in \Omega} \langle \mathbf{g} + \frac{1}{\gamma}(\mathbf{u}_t - \mathbf{u}), \mathbf{u}_t - \mathbf{x} \rangle$$

3: If $V_{\mathbf{g},\mathbf{u},\gamma}(\mathbf{u}_t) \leq \eta$, return $\mathbf{u}^+ = \mathbf{u}_t$.
4: Set $\mathbf{u}_{t+1} = (1 - \alpha_t)\mathbf{u}_t + \alpha_t \mathbf{v}_t$ where $\alpha_t = \min\{1, \frac{\langle \frac{1}{\gamma}(\mathbf{u}-\mathbf{u}_t)-\mathbf{g}, \mathbf{v}_t - \mathbf{u}_t \rangle}{\frac{1}{\gamma}\|\mathbf{v}_t - \mathbf{u}_t\|^2}\}$.
5: Set $t \leftarrow t + 1$ and goto step 2.

SGFFW: Stochastic Gradient-Free Frank-Wolfe

- It was one of the first approaches to Stochastic Zeroth-Order Frank-Wolfe
- Uses Random Directions Gradient Estimator
- Employs an averaging trick to counter diverging gradient
- Sampled Directions:
  - KWSA: along each coordinate direction (d directions, with d dimensionality of the problem space)
  - RDSA: one random direction
  - I-RDSA: m < d independently sampled directions

Algorithm 2 Stochastic Gradient Free Frank Wolfe

Require: Input, Loss Function $F(x)$, Convex Set $\mathcal{C}$, number of directions m, sequences $\gamma_t = \frac{2}{t+8}$,

$$(\rho_t, c_t)_{RDSA} = \left(\frac{4}{d^{1/3}(t+8)^{2/3}}, \frac{2}{d^{3/2}(t+8)^{1/3}}\right)$$

$$(\rho_t, c_t)_{I-RDSA} = \left(\frac{4}{\left(1+\frac{d}{m}\right)^{1/3}(t+8)^{2/3}}, \frac{2\sqrt{m}}{d^{3/2}(t+8)^{1/3}}\right)$$

$$(\rho_t, c_t)_{KWSA} = \left(\frac{4}{(t+8)^{2/3}}, \frac{2}{d^{1/2}(t+8)^{1/3}}\right).$$

Output: $\mathbf{x}_T$ or $\frac{1}{T}\sum_{t=0}^{T-1} \mathbf{x}_t$.

1: Initialize $\mathbf{x}_0 \in \mathcal{C}$
2: **for** $t = 0, 2, \ldots, T-1$ **do**
3: Compute
KWSA: $\mathbf{g}(\mathbf{x}_t; \mathbf{y}) = \sum_{i=1}^{d} \frac{F(\mathbf{x}_t + c_t \mathbf{e}_i; \mathbf{y}) - F(\mathbf{x}_t; \mathbf{y})}{c_t} \mathbf{e}_i$
RDSA: Sample $\mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I}_d)$,
$\mathbf{g}(\mathbf{x}_t; \mathbf{y}, \mathbf{z}_t) = \frac{F(\mathbf{x}_t + c_t \mathbf{z}_t; \mathbf{y}) - F(\mathbf{x}_t; \mathbf{y})}{c_t} \mathbf{z}_t$
I-RDSA: Sample $\{\mathbf{z}_{i,t}\}_{i=1}^{m} \sim \mathcal{N}(0, \mathbf{I}_d)$,
$\mathbf{g}(\mathbf{x}_t; \mathbf{y}, \mathbf{z}_t) = \frac{1}{m}\sum_{i=1}^{m} \frac{F(\mathbf{x}_t + c_t \mathbf{z}_{i,t}; \mathbf{y}) - F(\mathbf{x}_t; \mathbf{y})}{c_t} \mathbf{z}_{i,t}$
4: Compute $\mathbf{d}_t = (1-\rho_t)\,\mathbf{d}_{t-1} + \rho_t \mathbf{g}(\mathbf{x}_t, \mathbf{y}_t)$
5: Compute $\mathbf{v}_t = \mathrm{argmin}_{\mathbf{s} \in \mathcal{C}} \langle \mathbf{s}, \mathbf{d}_t) \rangle$,
6: Compute $\mathbf{x}_{t+1} = (1-\gamma_t)\,\mathbf{x}_t + \gamma_t \mathbf{v}_t$.
7: **end for**

Experiments

- The experiment conducted on these **Frank-Wolfe** variants is a Black-Box **Adversarial Attack** on the **MNIST Dataset**.
  - This means that we have to find the **smallest perturbation** of the input 28x28 greyscale image of a handwritten digit, constrained by $\|\delta\|_\infty \leq s$, able to make the Deep Neural Network take the incorrect prediction.

- We follow the **setup** from `nn-carlini` pre-trained DNN for the MNIST dataset.

- **Feasible Set** used:
  - As per section 4.3 of Gao et al., the feasible set used is $\|\delta\|_\infty \leq s$
  - *s* has different **values** based on the algorithm:
    - **FZCGS**: *s* = 4
    - **SGFFW** with **RDSA**: *s* = 8
    - **SGFFW** with **I-RDSA** and **KWSA**: *s* = 4

Changes from Theoretical Papers

- The implementative changes form the Theoretical Papers concern the **FZCGS algorithm** (Gao et al.). In particular, the **Conditional Gradient** Procedure.
- condg( ) from Gao et al. was **changed** with the condg( ) version from the paper Lobanov et al.
- condg( ) also presents an **additional stopping condition** based on the value of the **alpha** parameter.
  - This additional condition **speeds up** the already extremely **long running times** of FZCGS, whilst producing **negligible deviations** from the otherwise computed result.

Algorithm 3 $\mathbf{u}^+ = \text{condg}(\mathbf{g}, \mathbf{u}, \gamma, \eta)$ (Qu et al., 2017)

1: $\mathbf{u}_1 = \mathbf{u}$, $t = 1$
2: $\mathbf{v}_t$ be an optimal solution for

$$V_{\mathbf{g},\mathbf{u},\gamma}(\mathbf{u}_t) = \max_{\mathbf{x}\in\Omega} \langle \mathbf{g} + \frac{1}{\gamma}(\mathbf{u}_t - \mathbf{u}), \mathbf{u}_t - \mathbf{x} \rangle$$

3: If $V_{\mathbf{g},\mathbf{u},\gamma}(\mathbf{u}_t) \leq \eta$, return $\mathbf{u}^+ = \mathbf{u}_t$.
4: Set $\mathbf{u}_{t+1} = (1 - \alpha_t)\mathbf{u}_t + \alpha_t \mathbf{v}_t$ where $\alpha_t = \min\{1, \frac{\langle \frac{1}{\gamma}(\mathbf{u}-\mathbf{u}_t)-\mathbf{g}, \mathbf{v}_t-\mathbf{u}_t\rangle}{\frac{1}{\gamma}\|\mathbf{v}_t-\mathbf{u}_t\|^2}\}$.
5: Set $t \leftarrow t + 1$ and goto step 2.

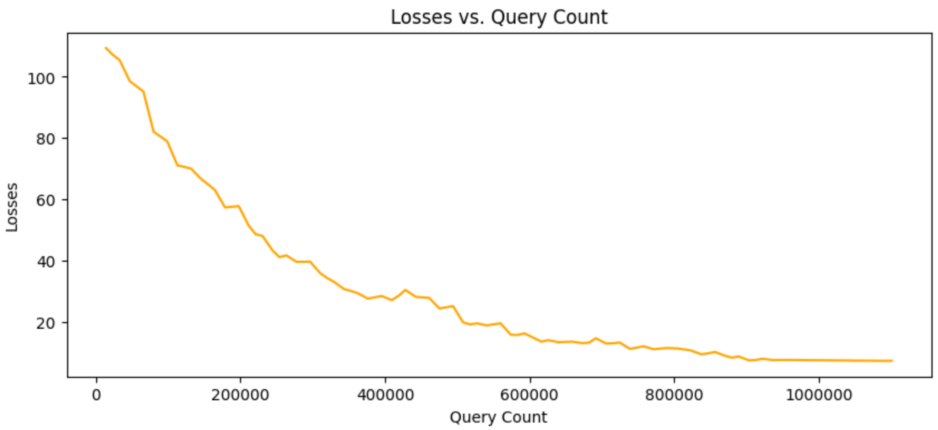
Conditional Gradient procedure (Gao et al.)

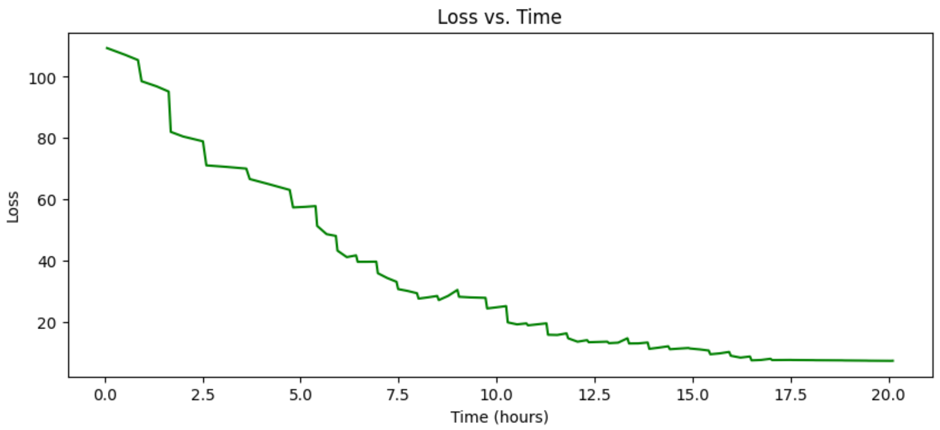
Algorithm 2 Conditional Gradient procedure

1: **for** $t = 0, ..., T$ **do**
2: $\quad v_t \leftarrow \text{argmin}_{v\in Q}\langle g_t, v\rangle$
3: $\quad$ **if** $\langle g_t, u_t - v_t\rangle \leq \beta$ **then**
4: $\quad\quad$ **return** u_t
5: $\quad$ **end if**
6: $\quad \alpha_t \leftarrow \min\left\{\frac{\langle g_t, u_t - v_t\rangle}{\eta\|u_t - v_t\|^2}, 1\right\}$
7: $\quad u_{t+1} \leftarrow u_t + \alpha_t(v_t - u_t)$
8: $\quad g_{t+1} \leftarrow g_0 + \eta(u_{t+1} - u_0)$
9: **end for**

Conditional Gradient procedure (Lobanov et al.)

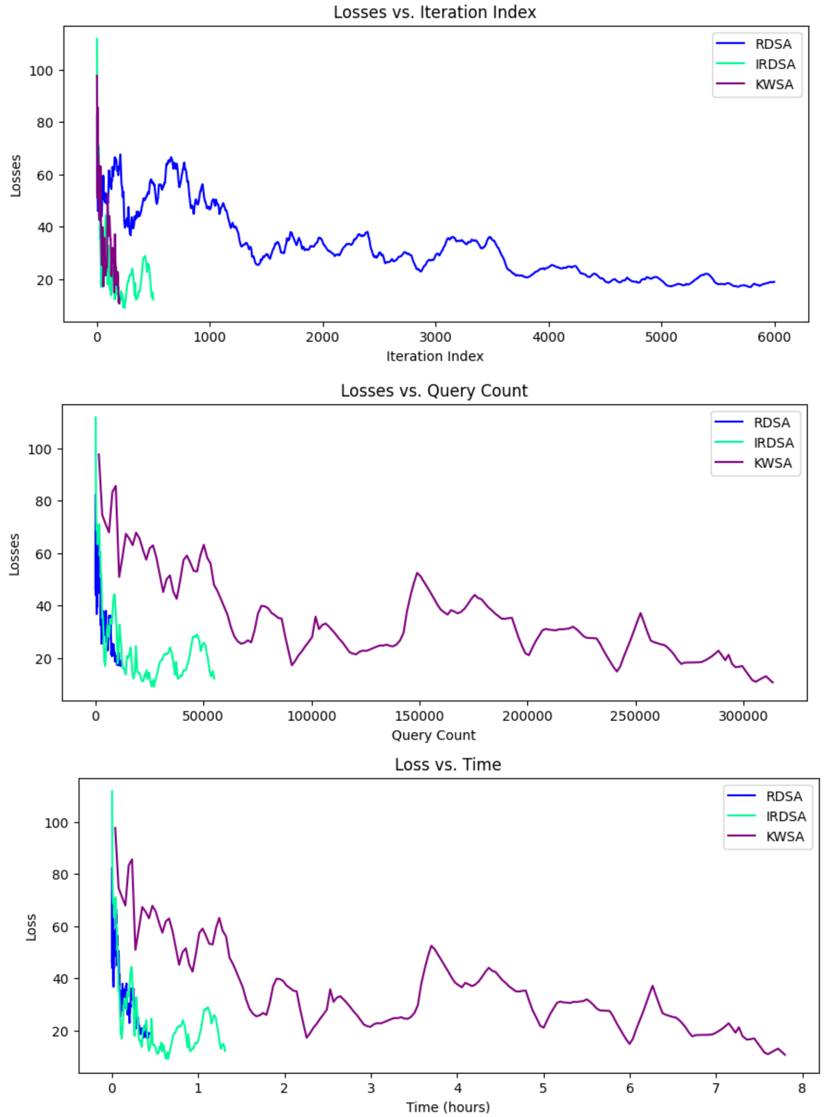
FZCGS Results





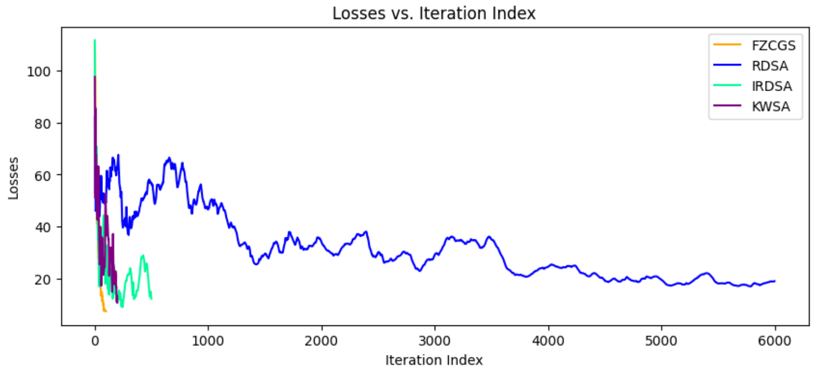
SGFFW Results (Compared)

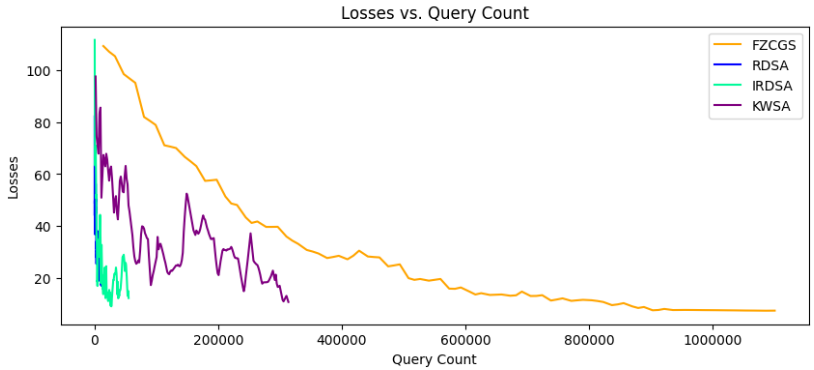
- **SGFFW** has different behaviours depending on which gradient approximation scheme is used.
- **RDSA**: fastest in time and with limited query count, while requiring a lot of iterations.
- **I-RDSA**: requires extremely less iterations than RDSA, but requires more queries.
- **KWSA**: slowest in time, requires the least amount of iterations, at the expense of a very large query count.
- **I-RDSA** shows a remarkable **compromise** between **iterations, query count** and **running time** to converge.

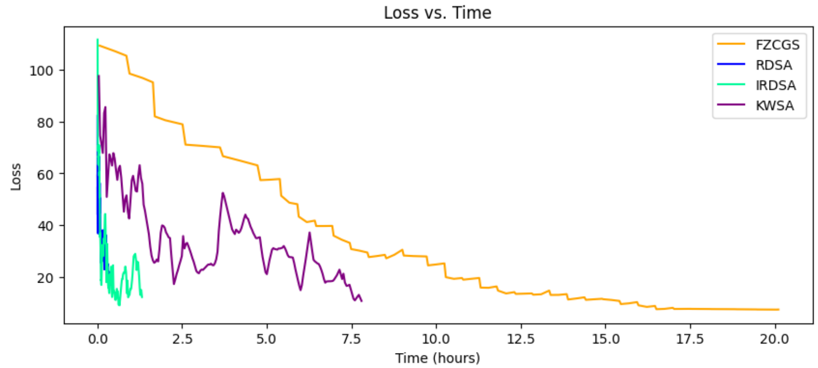


Results: All Algorithms Compared

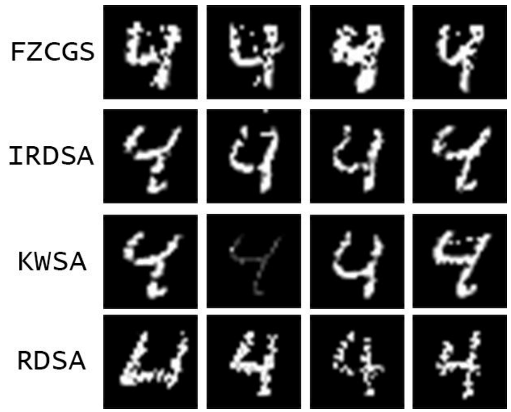
Algorithm	Metric	Value
FZCGS	Best Loss	7.283
	Last Loss	7.312
	Iteration Count	100
	Query Count	1100736
	Running Time (Hrs)	20.1
SGFFW-RDSA	Best Loss	16.9
	Last Loss	18.922
	Iteration Count	6000
	Query Count	12000
	Running Time (Hrs)	0.43
SGFFW-I-RDSA	Best Loss	8.96
	Last Loss	12.156
	Iteration Count	500
	Query Count	55000
	Running Time (Hrs)	1.3
SGFFW-KWSA	Best Loss	10.647
	Last Loss	10.647
	Iteration Count	200
	Query Count	313600
	Running Time (Hrs)	7.7







Conclusions



- There is **no overall best algorithm**, since each of them has **strengths** and **weaknesses**.
- **FZCGS** has the **best** performance for **iterations** after SGFFW with **KWSA**, but is the **worst** for query count and running times.
- **SGFFW with I-RDSA** shows a remarkable **compromise** between all metrics:
  - Requires an acceptable amount of **iterations**
  - The **query** count is reasonably small
  - Running **times** are convenient
  - Reaches one of the **lowest losses**