Randomized Experiment

```r
library(conflicted)

library(kableExtra)
library(knitr)
library(broom.helpers)
library(broom)
library(dtplyr)
library(furrr)
```

```
## Loading required package: future
```

```r
library(arrow)
library(glue)
library(fs)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
```

```r
conflict_prefer("filter", "dplyr")
```

```
## [conflicted] Will prefer dplyr::filter over any other package.
```

```r
source(here("analysis/utils.R"), local = knit_global())
set_theme()
```

```r
write_bib(.packages(), here("analysis/packages.bib"))
sessionInfo()
```

```
## R version 4.4.0 (2024-04-24)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.5
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  LAPACK v
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: Asia/Singapore
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
```

```
##
## other attached packages:
##  [1] lubridate_1.9.3      forcats_1.0.0        stringr_1.5.1
##  [4] dplyr_1.1.4          purrr_1.0.2          readr_2.1.5
##  [7] tidyr_1.3.1          tibble_3.2.1         ggplot2_3.5.1
## [10] tidyverse_2.0.0      fs_1.6.4             glue_1.7.0
## [13] arrow_16.1.0         furrr_0.3.1          future_1.33.2
## [16] dtplyr_1.3.1         broom_1.0.6          broom.helpers_1.15.0
## [19] knitr_1.47           kableExtra_1.4.0     conflicted_1.2.0
## [22] here_1.0.1
##
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.5      xfun_0.45         tzdb_0.4.0        vctrs_0.6.5
##  [5] tools_4.4.0       generics_0.1.3    parallel_4.4.0    fansi_1.0.6
##  [9] pkgconfig_2.0.3   data.table_1.15.4 assertthat_0.2.1  lifecycle_1.0.4
## [13] compiler_4.4.0    munsell_0.5.1     codetools_0.2-20  htmltools_0.5.8.1
## [17] yaml_2.3.8        pillar_1.9.0      cachem_1.1.0      parallelly_1.37.1
## [21] tidyselect_1.2.1  digest_0.6.35     stringi_1.8.4     listenv_0.9.1
## [25] rprojroot_2.0.4   fastmap_1.2.0     grid_4.4.0        colorspace_2.1-0
## [29] cli_3.6.2         magrittr_2.0.3    utf8_1.2.4        withr_3.0.0
## [33] scales_1.3.0      backports_1.5.0   bit64_4.0.5       timechange_0.3.0
## [37] rmarkdown_2.27    globals_0.16.3    bit_4.0.5         hms_1.1.3
## [41] memoise_2.0.1     evaluate_0.24.0   viridisLite_0.4.2 rlang_1.1.4
## [45] xml2_1.3.6        svglite_2.1.3     rstudioapi_0.16.0 R6_2.5.1
## [49] systemfonts_1.1.0
```

Analyze attack trends

```r
data_dir <- here(glue("{params$data}/{params$simulation}/results"))

success_fnames <-
  dir_ls(data_dir, glob = glue("*norm_{params$norm}*.csv"))

stopifnot(length(success_fnames) == 1200)

# every fname is a simulation
success_raw_data <- get_data(success_fnames, read_csv) |>
  glimpse()
```

```
## Rows: 1,200
## Columns: 16
## $ fname                 <chr> "/Users/zbli/Documents/Documents - ZhaoBin's M~
## $ num_iteration         <dbl> 100, 100, 100, 100, 100, 100, 100, 100, 100, 1~
## $ max_norm              <dbl> 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05~
## $ model_name            <ord> Cascade R-CNN, Faster R-CNN, RetinaNet, SSD, Y~
## $ loss_target           <ord> Mislabeling, Mislabeling, Mislabeling, Mislabe~
## $ attack_bbox           <chr> "predictions", "predictions", "predictions", "~
## $ perturb_fun           <chr> "perturb_inside", "perturb_inside", "perturb_i~
## $ sample_count          <dbl> 247, 253, 258, 266, 261, 247, 253, 258, 266, 2~
## $ attack_count          <dbl> 200, 200, 200, 200, 200, 200, 200, 200, 200, 2~
## $ success_count         <dbl> 4, 6, 4, 31, 42, 7, 8, 9, 28, 13, 5, 4, 11, 40~
## $ vanish_count          <dbl> 2, 5, 0, 11, 14, 7, 8, 6, 22, 10, 5, 4, 10, 39~
## $ mislabel_count        <dbl> 2, 1, 4, 20, 28, 0, 0, 3, 6, 3, 0, 0, 1, 1, 0,~
## $ mislabel_intended_count <dbl> 2, 1, 4, 20, 27, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ target_max_conf         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ perturb_min_size        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ bbox_max_dist           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
```

```
itr_lab <- "Attack Iterations"

cap <- glue("{emp_tex('Intent obfuscating attack is feasible for all models and attacks', params$norm)}

cap
```

```
## Intent obfuscating attack is feasible for all models and attacks even with 0.05 max-norm:  We conduc
```

```
success_intended_data <- success_raw_data |>
  mutate(success_intended_count = case_when(
    loss_target == "Mislabeling" ~ mislabel_intended_count,
    loss_target == "Vanishing" ~ vanish_count,
    loss_target == "Untargeted" ~ success_count
  ))

# expand intended success per simulation into 1 and 0s per row
success_expanded_data <- success_intended_data |>
  rowwise() |>
  mutate(success = list(rep(0:1, times = c(attack_count - success_intended_count, success_intended_coun
  unnest_longer(success) |>
  glimpse()
```

```
## Rows: 240,000
## Columns: 18
## $ fname                   <chr> "/Users/zbli/Documents/Documents - ZhaoBin's M~
## $ num_iteration           <dbl> 100, 100, 100, 100, 100, 100, 100, 100, 100, 1~
## $ max_norm                <dbl> 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05~
## $ model_name              <ord> Cascade R-CNN, Cascade R-CNN, Cascade R-CNN, C~
## $ loss_target             <ord> Mislabeling, Mislabeling, Mislabeling, Mislabe~
## $ attack_bbox             <chr> "predictions", "predictions", "predictions", "~
## $ perturb_fun             <chr> "perturb_inside", "perturb_inside", "perturb_i~
## $ sample_count            <dbl> 247, 247, 247, 247, 247, 247, 247, 247, 247, 2~
## $ attack_count            <dbl> 200, 200, 200, 200, 200, 200, 200, 200, 200, 2~
## $ success_count           <dbl> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4~
## $ vanish_count            <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ mislabel_count          <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ mislabel_intended_count <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ target_max_conf         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ perturb_min_size        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ bbox_max_dist           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ success_intended_count  <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ success                 <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
```

```
# use log(num_iteration)
g <- success_expanded_data |>
  ggplot(aes(num_iteration, success, color = loss_target, linetype = loss_target)) +
  # use stat_summary rather than stat_summary_bin
  # since num_iteration is set experimentally
  # mean_cl_boot gives 95% bootstrapped CI at 1000 samples
  # https://rdrr.io/cran/Hmisc/man/smean.sd.html
  stat_summary(fun.data = "mean_cl_boot") +
```

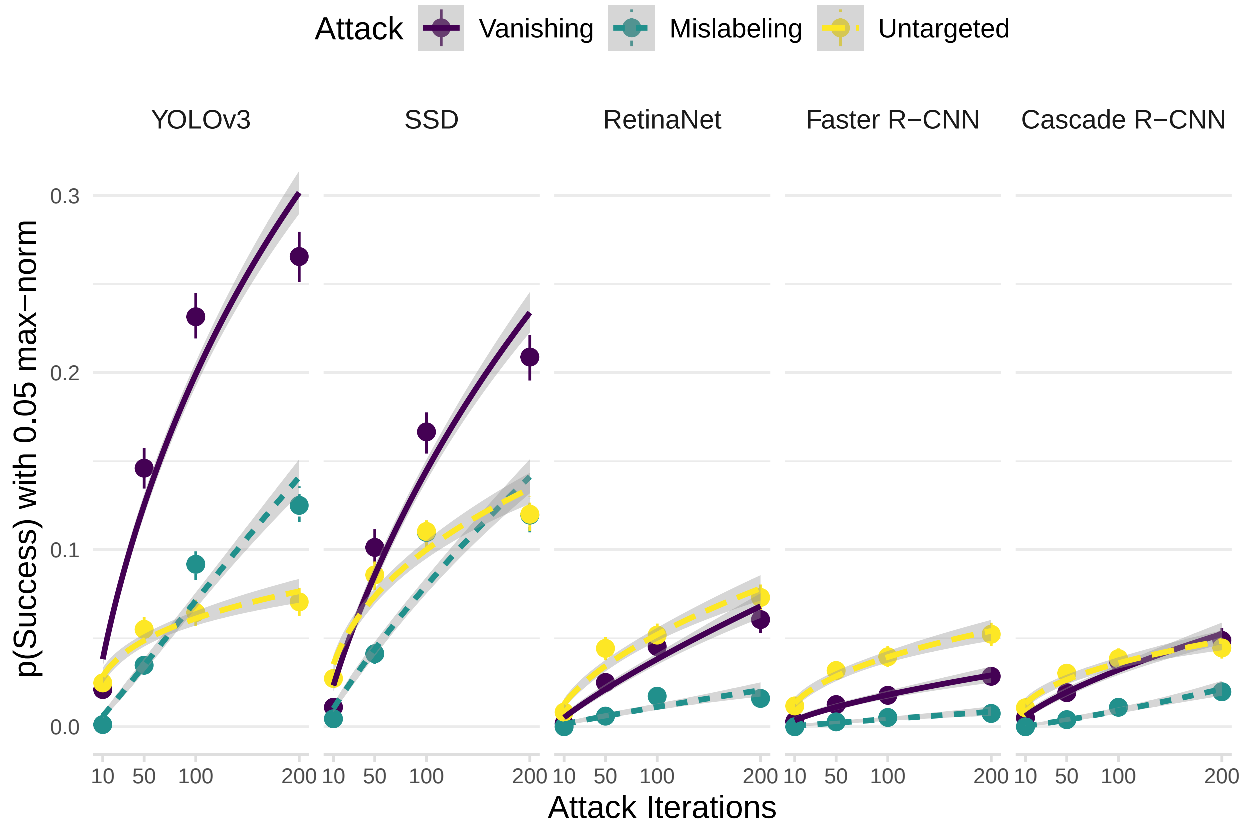


Figure 1: Intent obfuscating attack is feasible for all models and attacks even with 0.05 max-norm: We conduct a randomized experiment by resampling COCO images, and within those images randomly sampling correctly predicted target and perturb objects. Then we distort the perturb objects to disrupt the target objects varying the attack iterations. The binned summaries and regression trendlines graph success proportion against attack iterations in the randomized attack experiment. Errors are 95% confidence intervals and every point aggregates success over 4,000 images. Targeted vanishing and mislabeling attacks obtain significantly greater success on the 1-stage YOLOv3 and SSD than the 2-stage Faster R-CNN and Cascade R-CNN detectors. However, the 1-stage RetinaNet is as resilient as the 2-stage detectors. Moreover, success rates significantly increase with larger attack iterations. Significance is determined at $\alpha < 0.05$ using a Wald z-test on the logistic estimates. Full details are given in Section ??.

```
  binomial_smooth(formula = y ~ log(x)) +
  facet_grid(cols = vars(model_name))

g +
  labs(x = itr_lab, y = glue("p(Success) {norm_axy(params$norm)}"), color = "Attack", linetype = "Attac
  scale_x_continuous(breaks = unique(success_raw_data$num_iteration))
```

```
# compare models against YOLO
# grouped by attack
data <- success_expanded_data |>
  # restrict to max iteration
  filter(num_iteration == max(num_iteration)) |>
  # avoid ordered regression
  mutate(
    model_name = factor(model_name, ordered = FALSE),
    loss_target = factor(loss_target, ordered = FALSE)
  ) |>
  glimpse()
```

```
## Rows: 60,000
## Columns: 18
## $ fname                  <chr> "/Users/zbli/Documents/Documents - ZhaoBin's M~
## $ num_iteration          <dbl> 200, 200, 200, 200, 200, 200, 200, 200, 200, 2~
## $ max_norm               <dbl> 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05~
## $ model_name             <fct> Cascade R-CNN, Cascade R-CNN, Cascade R-CNN, C~
## $ loss_target            <fct> Mislabeling, Mislabeling, Mislabeling, Mislabe~
## $ attack_bbox            <chr> "predictions", "predictions", "predictions", "~
## $ perturb_fun            <chr> "perturb_inside", "perturb_inside", "perturb_i~
## $ sample_count           <dbl> 247, 247, 247, 247, 247, 247, 247, 247, 247, 2~
## $ attack_count           <dbl> 200, 200, 200, 200, 200, 200, 200, 200, 200, 2~
## $ success_count          <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8~
## $ vanish_count           <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ mislabel_count         <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6~
## $ mislabel_intended_count <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6~
## $ target_max_conf        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ perturb_min_size       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ bbox_max_dist          <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ success_intended_count <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6~
## $ success                <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
```

```
model <- partial(glm_model, predictor = "model_name")

reg_est <- get_tidied_reg(
  model, data, loss_target
)
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## `summarise()` has grouped output by 'loss_target'. You can override using the
## `.groups` argument.
```

```
ext_sig(reg_est)
```

```
## Total 15 predictors:
## 10 (67%) significant;
## 10 (67%) both

## # A tibble: 10 x 8
## # Groups:   loss_target [3]
##    loss_target term       estimate std.error statistic p.value conf.low conf.high
##    <fct>       <chr>         <dbl>     <dbl>     <dbl>   <dbl>    <dbl>     <dbl>
##  1 Vanishing   model_na~    -0.315     0.053     -5.96   0       -0.419    -0.211
##  2 Vanishing   model_na~    -1.72      0.075    -22.9    0       -1.88     -1.58
##  3 Vanishing   model_na~    -2.51      0.102    -24.7    0       -2.72     -2.32
##  4 Vanishing   model_na~    -1.95      0.082    -23.9    0       -2.12     -1.80
##  5 Mislabeling model_na~   -2.17      0.135    -16.1     0        -2.45     -1.92
##  6 Mislabeling model_na~   -2.94      0.189    -15.5     0        -3.33     -2.59
##  7 Mislabeling model_na~   -1.96      0.123    -15.9     0        -2.21     -1.72
##  8 Untargeted  model_na~    0.587     0.079      7.46    0         0.433     0.742
##  9 Untargeted  model_na~   -0.319     0.094     -3.39    0.001    -0.504    -0.135
## 10 Untargeted  model_na~   -0.488     0.098     -4.95    0        -0.682    -0.296
```

```r
cap <- table_caption("detection models, split by attack,", "Both vanishing and mislabeling attacks obta

print_statistics(reg_est, cap)
```

Table 1: We run a logistic model regressing success against detection models, split by attack, in the randomized attack experiment. Both vanishing and mislabeling attacks obtain higher success on 1-stage (YOLOv3, SSD) than 2-stage (Faster R-CNN, Cascade R-CNN) detectors. However, the 1-stage RetinaNet is as resilient as 2-stage detectors. Table headers are explained in Appendix **??**.

Group	Regression							
Attack	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
Vanishing	YOLOv3		0.000					
	SSD	*	-0.315	0.053	-5.956	0.000	-0.419	-0.211
	RetinaNet	*	-1.725	0.075	-22.889	0.000	-1.875	-1.579
	Faster R-CNN	*	-2.511	0.102	-24.732	0.000	-2.715	-2.317
	Cascade R-CNN	*	-1.953	0.082	-23.914	0.000	-2.116	-1.796
Mislabeling	YOLOv3		0.000					
	SSD		-0.051	0.068	-0.751	0.453	-0.185	0.083
	RetinaNet	*	-2.173	0.135	-16.124	0.000	-2.446	-1.917
	Faster R-CNN	*	-2.939	0.189	-15.521	0.000	-3.332	-2.587
	Cascade R-CNN	*	-1.959	0.123	-15.888	0.000	-2.207	-1.723
Untargeted	YOLOv3		0.000					
	SSD	*	0.587	0.079	7.460	0.000	0.433	0.742
	RetinaNet		0.038	0.087	0.433	0.665	-0.132	0.208
	Faster R-CNN	*	-0.319	0.094	-3.389	0.001	-0.504	-0.135
	Cascade R-CNN	*	-0.488	0.098	-4.954	0.000	-0.682	-0.296

```r
# compare attacks against vanishing
# grouped by models
model <- partial(glm_model, predictor = "loss_target")

reg_est <- get_tidied_reg(
  model, data, model_name
)
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## `summarise()` has grouped output by 'model_name'. You can override using the
## `.groups` argument.
```

```
ext_sig(reg_est)
```

```
## Total 15 predictors:
## 9 (60%) significant;
## 9 (60%) both

## # A tibble: 9 x 8
## # Groups:   model_name [5]
##   model_name    term     estimate std.error statistic p.value conf.low conf.high
##   <fct>         <chr>       <dbl>     <dbl>     <dbl>   <dbl>    <dbl>     <dbl>
## 1 YOLOv3        loss_ta~   -0.928     0.06     -15.5    0        -1.05     -0.812
## 2 YOLOv3        loss_ta~   -1.56      0.071    -21.9    0        -1.70     -1.42
## 3 SSD           loss_ta~   -0.665     0.062    -10.7    0        -0.787    -0.543
## 4 SSD           loss_ta~   -0.66      0.062    -10.6    0        -0.783    -0.538
## 5 RetinaNet     loss_ta~   -1.38      0.142     -9.67   0        -1.66     -1.10
## 6 RetinaNet     loss_ta~    0.201     0.09       2.24   0.025     0.025     0.378
## 7 Faster R-CNN  loss_ta~   -1.36      0.206     -6.57   0        -1.78     -0.966
## 8 Faster R-CNN  loss_ta~    0.631     0.119      5.32   0         0.401     0.866
## 9 Cascade R-CNN loss_ta~   -0.934     0.135     -6.90   0        -1.20     -0.673
```

```
cap <- table_caption("attacks, split by detection models", "Targeted attacks obtain higher success than

print_statistics(reg_est, cap)
```

Table 2: We run a logistic model regressing success against attacks, split by detection models in the randomized attack experiment. Targeted attacks obtain higher success than untargeted attacks on YOLOv3 and SSD. Within targeted attacks, vanishing attacks obtain higher success than mislabeling attacks on all models. Table headers are explained in Appendix ??.

Group	Regression							
Model	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
YOLOv3	Vanishing		0.000					
	Mislabeling	*	-0.928	0.060	-15.542	0.000	-1.046	-0.812
	Untargeted	*	-1.561	0.071	-21.871	0.000	-1.703	-1.423
SSD	Vanishing		0.000					
	Mislabeling	*	-0.665	0.062	-10.658	0.000	-0.787	-0.543
	Untargeted	*	-0.660	0.062	-10.594	0.000	-0.783	-0.538
RetinaNet	Vanishing		0.000					
	Mislabeling	*	-1.376	0.142	-9.667	0.000	-1.663	-1.104
	Untargeted	*	0.201	0.090	2.237	0.025	0.025	0.378
Faster R-CNN	Vanishing		0.000					
	Mislabeling	*	-1.356	0.206	-6.571	0.000	-1.778	-0.966
	Untargeted	*	0.631	0.119	5.317	0.000	0.401	0.866
	Vanishing		0.000					
	Mislabeling	*	-0.934	0.135	-6.901	0.000	-1.204	-0.673

Cascade R-CNN								
	Untargeted		-0.096	0.106	-0.901	0.367	-0.304	0.112

```r
# num_iteration
reg_est <- get_tidied_reg(
  partial(glm_model, predictor = "log(num_iteration)"),
  success_expanded_data,
)
```

```
## `summarise()` has grouped output by 'model_name', 'loss_target'. You can
## override using the `.groups` argument.
```

```r
ext_sig(reg_est, "pos")
```

```
## Total 15 predictors:
## 15 (100%) significant;
## 15 (100%) pos
```

```
## # A tibble: 15 x 9
## # Groups:   model_name, loss_target [15]
##    model_name    loss_target term  estimate std.error statistic p.value conf.low
##    <ord>         <ord>       <chr>    <dbl>     <dbl>     <dbl>   <dbl>    <dbl>
##  1 YOLOv3        Vanishing   log(~    0.797     0.027     29.7        0    0.745
##  2 YOLOv3        Mislabeling log(~    1.10      0.051     21.6        0    1
##  3 YOLOv3        Untargeted  log(~    0.347     0.036      9.62       0    0.277
##  4 SSD           Vanishing   log(~    0.852     0.032     26.6        0    0.79
##  5 SSD           Mislabeling log(~    0.922     0.044     20.9        0    0.837
##  6 SSD           Untargeted  log(~    0.483     0.031     15.7        0    0.423
##  7 RetinaNet     Vanishing   log(~    0.88      0.062     14.2        0    0.762
##  8 RetinaNet     Mislabeling log(~    0.903     0.115      7.86       0    0.688
##  9 RetinaNet     Untargeted  log(~    0.627     0.046     13.6        0    0.538
## 10 Faster R-CNN  Vanishing   log(~    0.707     0.082      8.66       0    0.552
## 11 Faster R-CNN  Mislabeling log(~    0.975     0.191      5.11       0    0.627
## 12 Faster R-CNN  Untargeted  log(~    0.483     0.049      9.94       0    0.389
## 13 Cascade R-CNN Vanishing   log(~    0.738     0.062     11.8        0    0.619
## 14 Cascade R-CNN Mislabeling log(~    1.25      0.149      8.40       0    0.972
## 15 Cascade R-CNN Untargeted  log(~    0.45      0.05       9.04       0    0.354
## # i 1 more variable: conf.high <dbl>
```

```r
cap <- table_caption(glue("log({itr_lab})"), "Success rates increase with attack iterations for all mode

print_statistics(reg_est, cap)
```

Table 3: We run a logistic model regressing success against log(attack iterations) in the randomized attack experiment. Success rates increase with attack iterations for all models and attacks. Table headers are explained in Appendix **??**.

Group	Regression							
Attack	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
YOLOv3								
Vanishing	log(iterations)	*	0.797	0.027	29.736	0	0.745	0.850
Mislabeling	log(iterations)	*	1.097	0.051	21.572	0	1.000	1.199
Untargeted	log(iterations)	*	0.347	0.036	9.615	0	0.277	0.419

SSD								
Vanishing	log(iterations)	*	0.852	0.032	26.573	0	0.790	0.915
Mislabeling	log(iterations)	*	0.922	0.044	20.885	0	0.837	1.010
Untargeted	log(iterations)	*	0.483	0.031	15.652	0	0.423	0.544
RetinaNet								
Vanishing	log(iterations)	*	0.880	0.062	14.229	0	0.762	1.005
Mislabeling	log(iterations)	*	0.903	0.115	7.855	0	0.688	1.139
Untargeted	log(iterations)	*	0.627	0.046	13.591	0	0.538	0.719
Faster R-CNN								
Vanishing	log(iterations)	*	0.707	0.082	8.664	0	0.552	0.872
Mislabeling	log(iterations)	*	0.975	0.191	5.111	0	0.627	1.378
Untargeted	log(iterations)	*	0.483	0.049	9.938	0	0.389	0.580
Cascade R-CNN								
Vanishing	log(iterations)	*	0.738	0.062	11.832	0	0.619	0.863
Mislabeling	log(iterations)	*	1.248	0.149	8.395	0	0.972	1.556
Untargeted	log(iterations)	*	0.450	0.050	9.040	0	0.354	0.549

Analyze individual cases

```r
# cache.lazy = FALSE needed to avoid errors with large bbox .parquets
attack_bbox <- "predictions"

bbox_fnames <-
  dir_ls(data_dir, glob = glue("*{params$norm}*.parquet"))

# Every bbox whether ground-truth, predicted or attacked is a row and the columns are the sample and bb
bbox_raw_data <- get_data(bbox_fnames, combine_trend_case) |>
  glimpse() |>
  lazy_dt()
```

```
## Rows: 8,712,402
## Columns: 41
## $ fname                      <chr> "/Users/zbli/Documents/Documents - ZhaoBin'~
## $ sample_id                  <chr> "65ed3a88141a475067f32706", "65ed3a88141a47~
## $ sample_path                <chr> "/projects/f_ps848_1/zhaobin/adversarial/co~
## $ sample_width               <int> 640, 640, 640, 640, 640, 640, 640, 640, 640~
## $ sample_height              <int> 480, 480, 480, 480, 480, 480, 480, 480, 480~
## $ sample_mislabel_class      <chr> "horse", "horse", "horse", "horse", "horse"~
## $ sample_mislabel_proba      <dbl> 6.615031e-05, 6.615031e-05, 6.615031e-05, 6~
## $ sample_attack              <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, T~
## $ sample_vanish              <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ sample_mislabel_intended   <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ sample_success             <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ sample_mislabel            <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ bbox_id                    <chr> "65ed3a88141a475067f32700", "65ed3a88141a47~
## $ bbox_class                 <chr> "clock", "person", "person", "person", "per~
## $ bbox_xywhn                 <list<double>> <0.32484375, 0.26458333, 0.0474218~
## $ bbox_conf                  <dbl> NA, NA, NA, NA, NA, NA, 0.9890913, 0.986363~
## $ bbox_res_eval              <chr> "tp", "tp", "tp", "tp", "tp", "fn", "tp", "~
## $ bbox_iou_eval              <dbl> 0.8860679, 0.8505562, 0.8757091, 0.8901640,~
## $ bbox_res_pgd_eval          <chr> NA, NA, NA, NA, NA, NA, "tp", "tp", "tp", "~
## $ bbox_iou_pgd_eval          <dbl> NA, NA, NA, NA, NA, NA, 0.9999464, 0.999894~
## $ bbox_res_pgd_mislabel_eval <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ bbox_iou_pgd_mislabel_eval <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ bbox_target                <lgl> TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FA~
## $ bbox_perturb               <lgl> FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FA~
## $ bbox_type                  <chr> "ground_truth", "ground_truth", "ground_tru~
## $ bbox_mislabel              <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ num_iteration              <dbl> 100, 100, 100, 100, 100, 100, 100, 100, 100~
## $ max_norm                   <dbl> 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0~
## $ model_name                 <ord> Cascade R-CNN, Cascade R-CNN, Cascade R-CNN~
## $ loss_target                <ord> Mislabeling, Mislabeling, Mislabeling, Misl~
## $ attack_bbox                <chr> "predictions", "predictions", "predictions"~
## $ perturb_fun                <chr> "perturb_inside", "perturb_inside", "pertur~
## $ sample_count               <dbl> 247, 247, 247, 247, 247, 247, 247, 247, 247~
## $ attack_count               <dbl> 200, 200, 200, 200, 200, 200, 200, 200, 200~
## $ success_count              <dbl> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4~
## $ vanish_count               <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ mislabel_count             <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ mislabel_intended_count    <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ target_max_conf            <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ perturb_min_size           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ bbox_max_dist              <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
```

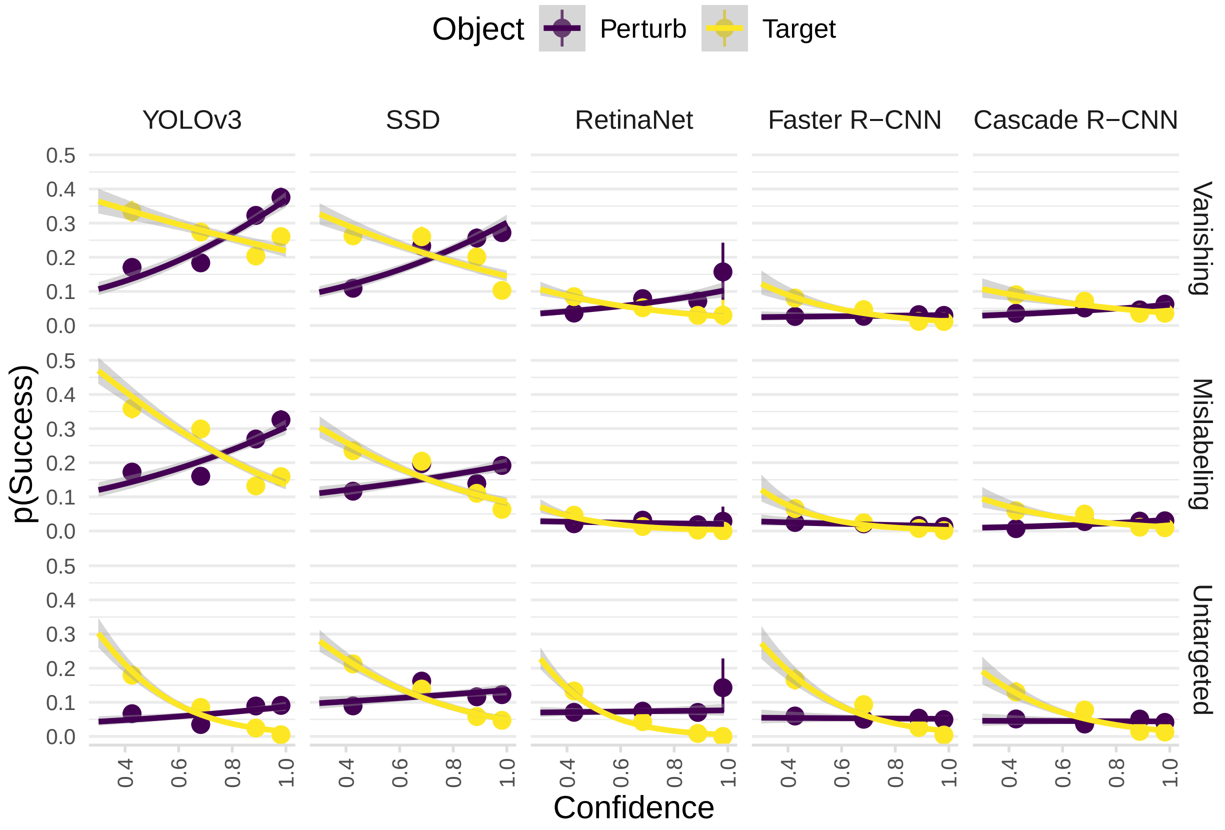
```r
# check whether target and perturb bboxes and
# mislabel classes are seeded across iterations
cols_start_equal(bbox_raw_data, c(
  "bbox_target", "bbox_perturb",
  "sample_mislabel_class", "sample_mislabel_proba"
))
```

```
## Columns starting with `bbox_target` are equal: TRUE
## Columns starting with `bbox_perturb` are equal: TRUE
## Columns starting with `sample_mislabel_class` are equal: TRUE
## Columns starting with `sample_mislabel_proba` are equal: TRUE
```

```r
# bbox confidence always based on predicted bbox
bbox_conf_data <- bbox_raw_data |>
  filter(bbox_type == "predictions") |>
  wrangle_success() |>
  glimpse()
```

```
## Rows: 120,000
## Columns: 42
## $ fname                      <chr> "/Users/zbli/Documents/Documents - ZhaoBin'~
## $ sample_id                  <chr> "65ed3a88141a475067f32706", "65ed3a88141a47~
## $ sample_path                <chr> "/projects/f_ps848_1/zhaobin/adversarial/co~
## $ sample_width               <int> 640, 640, 500, 640, 480, 640, 640, 640, 640~
## $ sample_height              <int> 480, 427, 332, 425, 640, 480, 480, 480, 640~
## $ sample_mislabel_class      <chr> "horse", "truck", "surfboard", "horse", "ca~
## $ sample_mislabel_proba      <dbl> 6.615031e-05, 4.219168e-02, 4.392489e-05, 1~
## $ sample_attack              <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, T~
## $ sample_vanish              <lgl> FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, FA~
## $ sample_mislabel_intended   <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ sample_success             <lgl> FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, FA~
## $ sample_mislabel            <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ bbox_id                    <chr> "65ed3aa3141a475067f3ca3e", "65ed3aa3141a47~
## $ bbox_class                 <chr> "clock", "car", "person", "person", "donut"~
## $ bbox_xywhn                 <list<double>> <0.32723613, 0.26601949, 0.0435188~
## $ bbox_conf                  <dbl> 0.9305881, 0.3433506, 0.9882318, 0.9988949,~
## $ bbox_res_eval              <chr> "tp", "tp", "tp", "tp", "tp", "tp", "tp", "~
## $ bbox_iou_eval              <dbl> 0.8860679, 0.7609860, 0.9454082, 0.9299325,~
## $ bbox_res_pgd_eval          <chr> "tp", "tp", "fn", "tp", "tp", "tp", "tp", "~
## $ bbox_iou_pgd_eval          <dbl> 1.0000000, 1.0000000, NA, 0.9999969, 1.0000~
## $ bbox_res_pgd_mislabel_eval <chr> NA, NA, "fn", NA, NA, NA, NA, NA, NA, NA, N~
## $ bbox_iou_pgd_mislabel_eval <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ bbox_type                  <chr> "predictions", "predictions", "predictions"~
## $ bbox_mislabel              <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ num_iteration              <dbl> 200, 200, 200, 200, 200, 200, 200, 200, 200~
## $ max_norm                   <dbl> 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0~
## $ model_name                 <ord> Cascade R-CNN, Cascade R-CNN, Cascade R-CNN~
## $ loss_target                <ord> Mislabeling, Mislabeling, Mislabeling, Misl~
## $ attack_bbox                <chr> "predictions", "predictions", "predictions"~
## $ perturb_fun                <chr> "perturb_inside", "perturb_inside", "pertur~
## $ sample_count               <dbl> 247, 247, 247, 247, 247, 247, 247, 247, 247~
## $ attack_count               <dbl> 200, 200, 200, 200, 200, 200, 200, 200, 200~
## $ success_count              <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8~
## $ vanish_count               <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ mislabel_count             <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6~
## $ mislabel_intended_count    <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6~
## $ target_max_conf            <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ perturb_min_size           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ bbox_max_dist              <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ target_or_perturb          <ord> Target, Target, Target, Target, Target, Tar~
## $ target_or_perturb_boolean  <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, T~
## $ success                    <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
```

```
bbox_conf_data |>
  graph_attr(bbox_conf, "Confidence")
```



```r
# restrict to target
pred_name <- "target confidence"
main_pt <- glue("Lower {pred_name} significantly increases success rates for all models and attacks")

bbox_conf_graph <- bbox_conf_data |> filter(target_or_perturb == "Target")
bbox_conf_graph |>
  graph_attr(bbox_conf, pred_name)
```

```r
model <- partial(glm_model, predictor = "bbox_conf")
data <- bbox_conf_graph

reg_est <- get_tidied_reg(model, data)
```

```
## `summarise()` has grouped output by 'model_name', 'loss_target'. You can
## override using the `.groups` argument.
```

```r
ext_sig(reg_est, "neg")
```

```
## Total 15 predictors:
## 15 (100%) significant;
## 15 (100%) neg
```

```
## # A tibble: 15 x 9
## # Groups:   model_name, loss_target [15]
##    model_name    loss_target term  estimate std.error statistic p.value conf.low
##    <ord>         <ord>       <chr>    <dbl>     <dbl>     <dbl>   <dbl>    <dbl>
##  1 YOLOv3        Vanishing   bbox~    -1.02     0.162     -6.29       0    -1.33
##  2 YOLOv3        Mislabeling bbox~    -2.47     0.171    -14.4        0    -2.81
##  3 YOLOv3        Untargeted  bbox~    -4.84     0.313    -15.5        0    -5.47
```

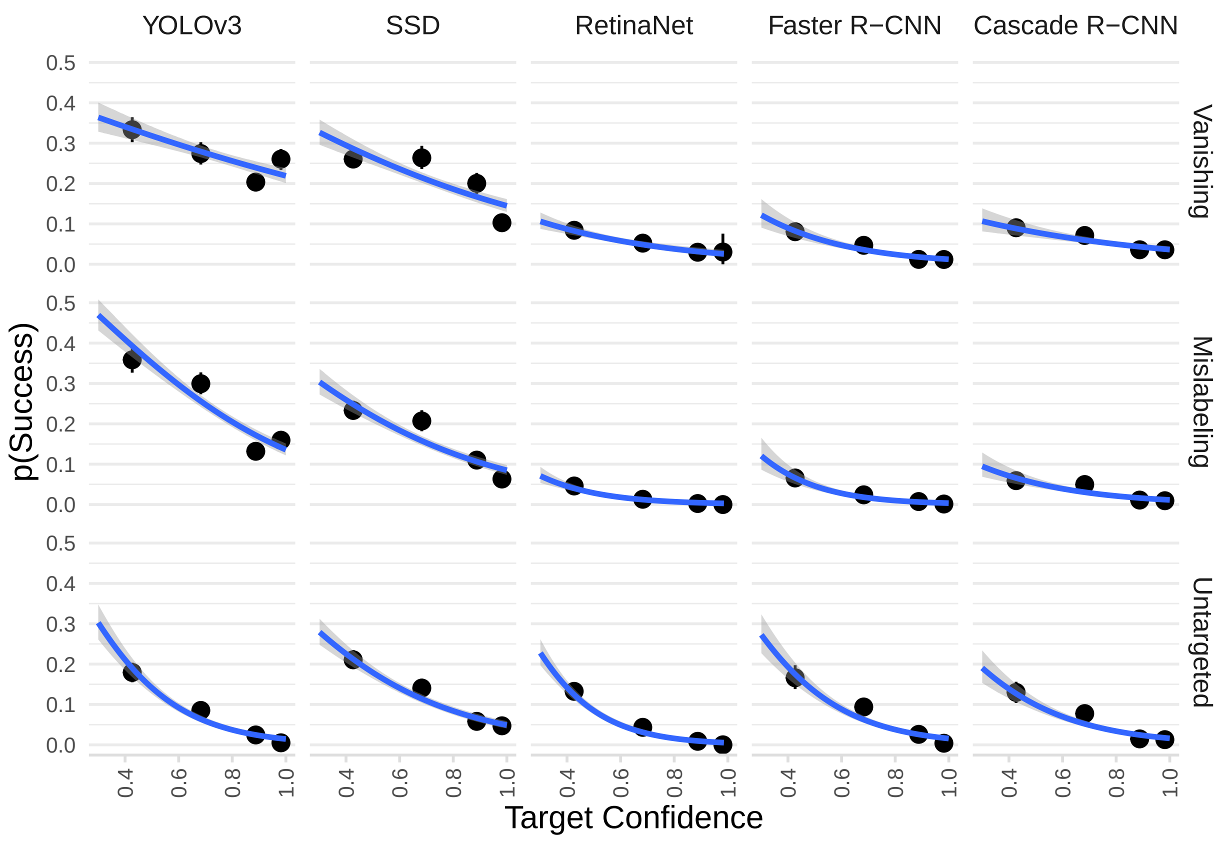


Figure 2: Lower target confidence significantly increases success rates for all models and attacks even with 0.05 max-norm: The binned summaries and regression trendlines graph success proportion against target confidence in the randomized attack experiment. Bins are split into quantiles. Errors are 95% confidence intervals

```
##  4 SSD           Vanishing   bbox~    -1.50     0.163     -9.25        0    -1.82
##  5 SSD           Mislabeling bbox~    -2.21     0.185    -12.0         0    -2.58
##  6 SSD           Untargeted  bbox~    -2.89     0.215    -13.5         0    -3.31
##  7 RetinaNet     Vanishing   bbox~    -2.20     0.36      -6.12        0    -2.92
##  8 RetinaNet     Mislabeling bbox~    -4.78     0.682     -7.00        0    -6.17
##  9 RetinaNet     Untargeted  bbox~    -5.82     0.439    -13.2         0    -6.70
## 10 Faster R-CNN  Vanishing   bbox~    -3.44     0.39      -8.81        0    -4.21
## 11 Faster R-CNN  Mislabeling bbox~    -5.24     0.56      -9.36        0    -6.38
## 12 Faster R-CNN  Untargeted  bbox~    -4.52     0.313    -14.4         0    -5.14
## 13 Cascade R-CNN Vanishing   bbox~    -1.65     0.303     -5.43        0    -2.24
## 14 Cascade R-CNN Mislabeling bbox~    -3.15     0.412     -7.64        0    -3.96
## 15 Cascade R-CNN Untargeted  bbox~    -3.81     0.326    -11.7         0    -4.46
## # i 1 more variable: conf.high <dbl>
```

```
print_statistics(reg_est, table_caption(pred_name, main_pt))
```

Table 4: We run a logistic model regressing success against target confidence in the randomized attack experiment. Lower target confidence significantly increases success rates for all models and attacks. Table headers are explained in Appendix ??.

Group	Regression

Attack	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
YOLOv3								
Vanishing	confidence	*	-1.017	0.162	-6.286	0	-1.334	-0.700
Mislabeling	confidence	*	-2.470	0.171	-14.445	0	-2.806	-2.136
Untargeted	confidence	*	-4.845	0.313	-15.476	0	-5.470	-4.241
SSD								
Vanishing	confidence	*	-1.505	0.163	-9.251	0	-1.825	-1.187
Mislabeling	confidence	*	-2.212	0.185	-11.970	0	-2.576	-1.852
Untargeted	confidence	*	-2.889	0.215	-13.462	0	-3.313	-2.471
RetinaNet								
Vanishing	confidence	*	-2.203	0.360	-6.124	0	-2.918	-1.507
Mislabeling	confidence	*	-4.778	0.682	-7.002	0	-6.173	-3.491
Untargeted	confidence	*	-5.816	0.439	-13.241	0	-6.701	-4.977
Faster R-CNN								
Vanishing	confidence	*	-3.442	0.390	-8.814	0	-4.213	-2.680
Mislabeling	confidence	*	-5.244	0.560	-9.361	0	-6.383	-4.178
Untargeted	confidence	*	-4.522	0.313	-14.433	0	-5.144	-3.915
Cascade R-CNN								
Vanishing	confidence	*	-1.647	0.303	-5.433	0	-2.237	-1.047
Mislabeling	confidence	*	-3.146	0.412	-7.635	0	-3.960	-2.341
Untargeted	confidence	*	-3.811	0.326	-11.692	0	-4.456	-3.177

```r
perturb_error_data <- bbox_conf_data |>
  filter(target_or_perturb == "Perturb") |>
  group_by(model_name, loss_target) |>
  summarise(perturb_error = 1 - mean(success)) |>
  glimpse()
```

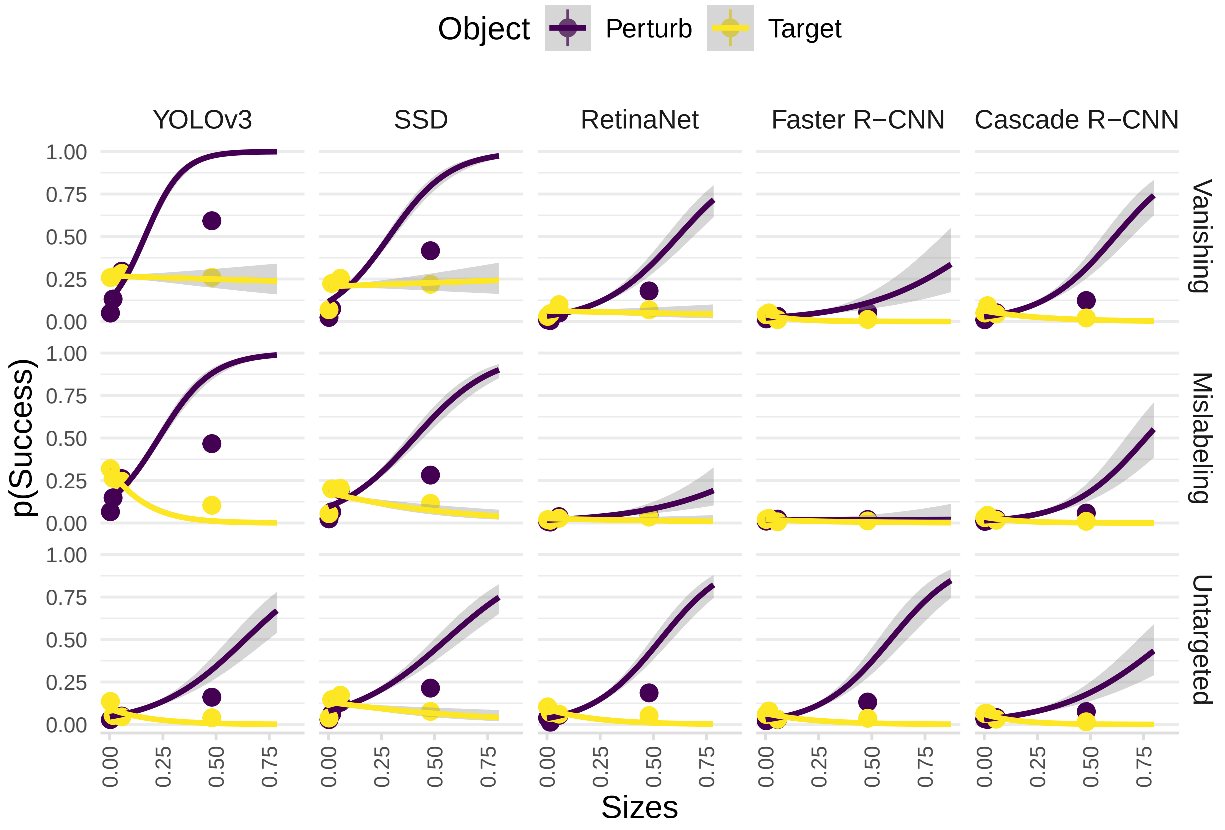
```
## `summarise()` has grouped output by 'model_name'. You can override using the
## `.groups` argument.
```

```
## Rows: 15
## Columns: 3
## Groups: model_name [5]
## $ model_name    <ord> YOLOv3, YOLOv3, YOLOv3, SSD, SSD, SSD, RetinaNet, Retina~
## $ loss_target   <ord> Vanishing, Mislabeling, Untargeted, Vanishing, Mislabeli~
## $ perturb_error <dbl> 0.73450, 0.76650, 0.92950, 0.79000, 0.84200, 0.88000, 0.~
```

```r
# bbox sizes typically based on ground-truth attacked bbox
bbox_size_data <- bbox_raw_data |>
  filter(bbox_type == attack_bbox) |>
  wrangle_success() |>
  # hoist not implemented in dtplyr
  as_tibble() |>
  # bbox_xywhn == normalized x1, y1, w, h
  hoist(bbox_xywhn, bbox_xn = 1, bbox_yn = 2, bbox_wn = 3, bbox_hn = 4) |>
  mutate(
    bbox_size = bbox_wn * bbox_hn,
  ) |>
  glimpse()
```

```
## Rows: 120,000
## Columns: 46
## $ fname                      <chr> "/Users/zbli/Documents/Documents - ZhaoBin'~
## $ sample_id                  <chr> "65ed3a88141a475067f32706", "65ed3a88141a47~
## $ sample_path                <chr> "/projects/f_ps848_1/zhaobin/adversarial/co~
## $ sample_width               <int> 640, 640, 500, 640, 480, 640, 640, 640, 640~
## $ sample_height              <int> 480, 427, 332, 425, 640, 480, 480, 480, 640~
## $ sample_mislabel_class      <chr> "horse", "truck", "surfboard", "horse", "ca~
## $ sample_mislabel_proba      <dbl> 6.615031e-05, 4.219168e-02, 4.392489e-05, 1~
## $ sample_attack              <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, T~
## $ sample_vanish              <lgl> FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, FA~
## $ sample_mislabel_intended   <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ sample_success             <lgl> FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, FA~
## $ sample_mislabel            <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ bbox_id                    <chr> "65ed3aa3141a475067f3ca3e", "65ed3aa3141a47~
## $ bbox_class                 <chr> "clock", "car", "person", "person", "donut"~
## $ bbox_xn                    <dbl> 0.32723613, 0.81016169, 0.37364487, 0.58023~
## $ bbox_yn                    <dbl> 0.26601949, 0.50290289, 0.31231453, 0.46766~
## $ bbox_wn                    <dbl> 0.04351888, 0.03631706, 0.35480569, 0.08531~
## $ bbox_hn                    <dbl> 0.10756386, 0.02172394, 0.67813552, 0.40265~
## $ bbox_conf                  <dbl> 0.9305881, 0.3433506, 0.9882318, 0.9988949,~
## $ bbox_res_eval              <chr> "tp", "tp", "tp", "tp", "tp", "tp", "tp", "~
## $ bbox_iou_eval              <dbl> 0.8860679, 0.7609860, 0.9454082, 0.9299325,~
## $ bbox_res_pgd_eval          <chr> "tp", "tp", "fn", "tp", "tp", "tp", "tp", "~
## $ bbox_iou_pgd_eval          <dbl> 1.0000000, 1.0000000, NA, 0.9999969, 1.0000~
## $ bbox_res_pgd_mislabel_eval <chr> NA, NA, "fn", NA, NA, NA, NA, NA, NA, NA, N~
## $ bbox_iou_pgd_mislabel_eval <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ bbox_type                  <chr> "predictions", "predictions", "predictions"~
## $ bbox_mislabel              <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ num_iteration              <dbl> 200, 200, 200, 200, 200, 200, 200, 200, 200~
## $ max_norm                   <dbl> 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0~
## $ model_name                 <ord> Cascade R-CNN, Cascade R-CNN, Cascade R-CNN~
## $ loss_target                <ord> Mislabeling, Mislabeling, Mislabeling, Misl~
## $ attack_bbox                <chr> "predictions", "predictions", "predictions"~
## $ perturb_fun                <chr> "perturb_inside", "perturb_inside", "pertur~
## $ sample_count               <dbl> 247, 247, 247, 247, 247, 247, 247, 247, 247~
## $ attack_count               <dbl> 200, 200, 200, 200, 200, 200, 200, 200, 200~
## $ success_count              <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8~
## $ vanish_count               <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ mislabel_count             <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6~
## $ mislabel_intended_count    <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6~
## $ target_max_conf            <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ perturb_min_size           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ bbox_max_dist              <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ target_or_perturb          <ord> Target, Target, Target, Target, Target, Tar~
## $ target_or_perturb_boolean  <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, T~
## $ success                    <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ bbox_size                  <dbl> 0.0046810584, 0.0007889497, 0.2406063427, 0~
```

```r
bbox_size_data |>
  graph_attr(bbox_size, "Sizes")
```

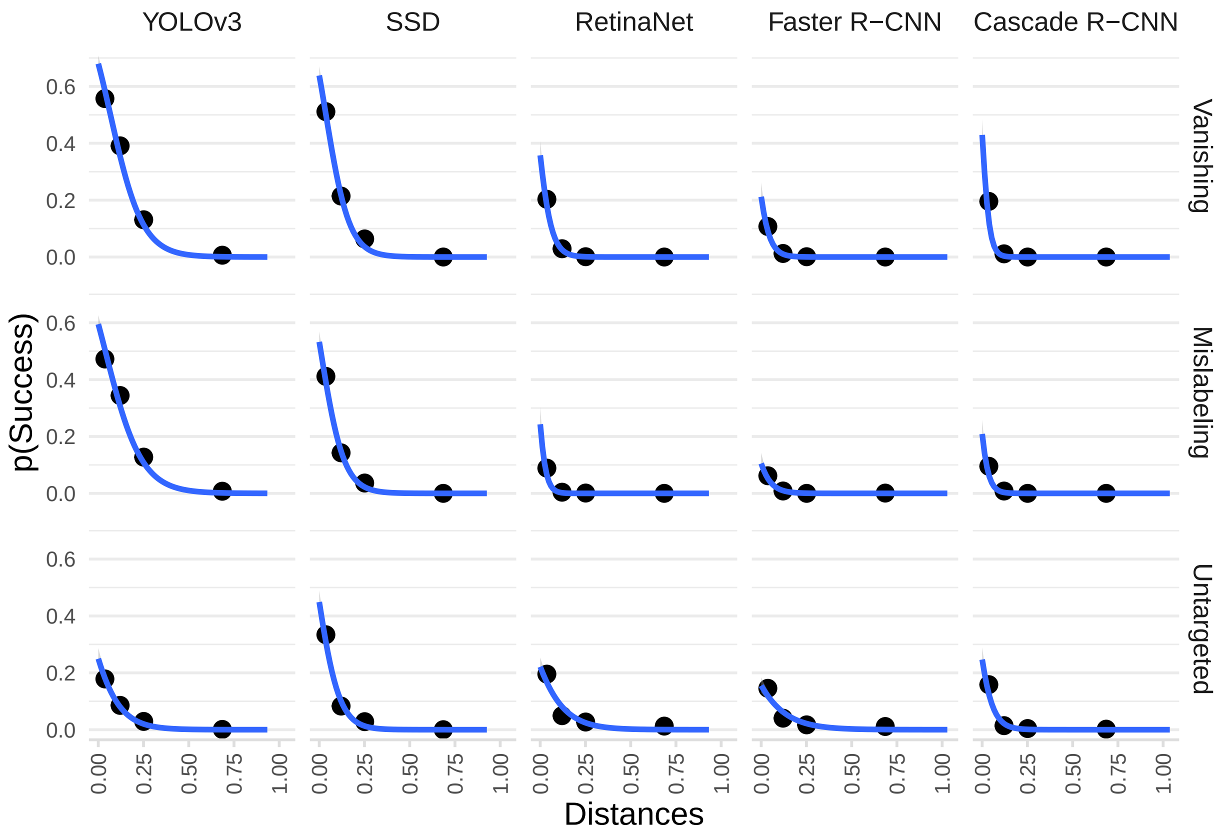


```r
# bbox distances typically based on ground-truth attacked bbox as in sizes
bbox_dist_data <- bbox_size_data |>
  mutate(
    target_or_perturb_lower = str_to_lower(target_or_perturb)
  ) |>
  # mainly "group" by sample_id and attack iteration
  # with target bbox on one row and perturb on another
  # success, model_name, loss_target are sample attributes
  # duplicated across bboxes
  pivot_wider(
    id_cols = c(fname, sample_id, num_iteration, success, model_name, loss_target), names_from = target_
    values_from = c(bbox_xn, bbox_yn, bbox_wn, bbox_hn, bbox_size)
  ) |>
  rowwise() |>
  mutate(bbox_dist = get_min_distance(
    bbox_xn_perturb, bbox_yn_perturb, bbox_xn_perturb + bbox_wn_perturb, bbox_yn_perturb + bbox_hn_pertu
    bbox_xn_target, bbox_yn_target, bbox_xn_target + bbox_wn_target, bbox_yn_target + bbox_hn_target
  )) |>
  ungroup() |>
  glimpse()
```

```
## Rows: 60,000
## Columns: 17
## $ fname              <chr> "/Users/zbli/Documents/Documents - ZhaoBin's MacBook~
## $ sample_id          <chr> "65ed3a88141a475067f32706", "65ed3a88141a475067f3272~
## $ num_iteration      <dbl> 200, 200, 200, 200, 200, 200, 200, 200, 200, 200, 20~
## $ success            <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ model_name         <ord> Cascade R-CNN, Cascade R-CNN, Cascade R-CNN, Cascade~
## $ loss_target       <ord> Mislabeling, Mislabeling, Mislabeling, Mislabeling, ~
## $ bbox_xn_target    <dbl> 0.32723613, 0.81016169, 0.37364487, 0.58023462, 0.82~
## $ bbox_xn_perturb   <dbl> 4.478896e-01, 1.517359e-01, 3.132355e-02, 2.802266e-~
## $ bbox_yn_target    <dbl> 0.26601949, 0.50290289, 0.31231453, 0.46766415, 0.18~
## $ bbox_yn_perturb   <dbl> 0.8013828, 0.5229044, 0.7769909, 0.4782841, 0.469777~
## $ bbox_wn_target    <dbl> 0.04351888, 0.03631706, 0.35480569, 0.08531094, 0.07~
## $ bbox_wn_perturb   <dbl> 0.02720404, 0.07043431, 0.18098172, 0.13681064, 0.12~
## $ bbox_hn_target    <dbl> 0.10756386, 0.02172394, 0.67813552, 0.40265309, 0.04~
## $ bbox_hn_perturb   <dbl> 0.07742354, 0.04831026, 0.21702971, 0.48990981, 0.04~
## $ bbox_size_target  <dbl> 0.0046810584, 0.0007889497, 0.2406063427, 0.03435071~
## $ bbox_size_perturb <dbl> 0.0021062328, 0.0034026994, 0.0392784101, 0.06702487~
## $ bbox_dist         <dbl> 0.43469769, 0.58799145, 0.16133960, 0.16319737, 0.28~
```

```
bbox_dist_data |>
  graph_attr(bbox_dist, "Distances")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```



```
saveRDS(bbox_dist_data, here(glue("analysis/rand_dist_size_norm_{params$norm}.RDS")))
```

```
check_graph_data(bbox_dist_data, c(bbox_dist, bbox_size_perturb))

dist_lab <- "Perturb-Target Distance (relative to image width/height)"
size_lab <- "Perturb Box Size (relative to image width/height)"

pred_name <- glue("{dist_lab} and {size_lab}")
main_pt <- "Larger perturb objects significantly increase success rates for all models and attacks, exc
```

```r
cap <- glue(
  "{emp_tex(main_pt, params$norm)} The binned summaries",
  " graph success proportion against {str_to_lower(pred_name)} in the",
  " randomized attack experiment."
)

bbox_dist_data <- bbox_dist_data |> mutate(
  bbox_size_perturb = bbox_size_perturb,
  bbox_dist = bbox_dist
)

graph_dist_size <- function(g) {
  g + facet_grid(rows = vars(loss_target), cols = vars(model_name)) +
    labs(x = dist_lab, y = size_lab) +
    scale_fill_viridis_c(name = "p(Success)", breaks = c(0, .5, 1), limits = c(0, 1))
}

g <- bbox_dist_data |> ggplot(aes(bbox_dist, bbox_size_perturb, z = success)) +
  stat_summary_2d(fun = "mean", bins = 5)

graph_dist_size(g)
```

```r
# control both
model <- partial(glm_model, predictor = "bbox_dist * bbox_size_perturb")
data <- bbox_dist_data

reg_res <- get_tidied_reg(model, data, return_mod = TRUE) |> glimpse()
```

```
## Warning: There were 4 warnings in `mutate()`.
## The first warning was:
## i In argument: `mod = list(model(data))`.
## i In row 7.
## Caused by warning:
## ! glm.fit: fitted probabilities numerically 0 or 1 occurred
## i Run `dplyr::last_dplyr_warnings()` to see the 3 remaining warnings.

## Warning: There were 168 warnings in `summarize()`.
## The first warning was:
## i In argument: `tidy_plus_plus(mod, conf.int = TRUE)`.
## i In row 7.
## Caused by warning:
## ! glm.fit: fitted probabilities numerically 0 or 1 occurred
## i Run `dplyr::last_dplyr_warnings()` to see the 167 remaining warnings.

## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## `summarise()` has grouped output by 'model_name', 'loss_target'. You can
## override using the `.groups` argument.
```

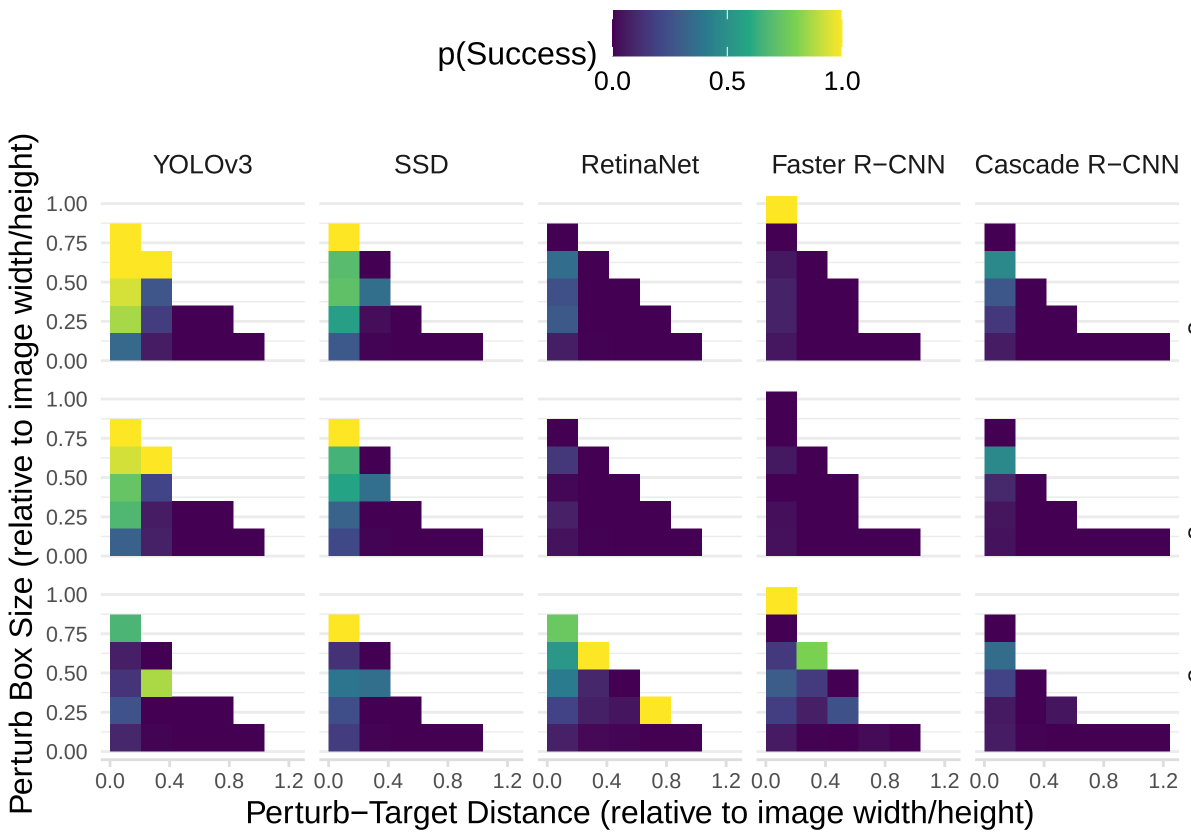


Figure 3: Larger perturb objects significantly increase success rates for all models and attacks, except for mislabeling attack on Faster R-CNN, after controlling for perturb-target distances. Shorter perturb-target distances significantly increase success rates for all models and attacks, after controlling for perturb object sizes even with 0.05 max-norm: The binned summaries graph success proportion against perturb-target distance (relative to image width/height) and perturb box size (relative to image width/height) in the randomized attack experiment.

```
## List of 2
##  $ mod   : rowws_df [15 x 4] (S3: rowwise_df/tbl_df/tbl/data.frame)
##   ..$ model_name : Ord.factor w/ 5 levels "YOLOv3"<"SSD"<..: 1 1 1 2 2 2 3 3 3 4 ...
##   ..$ loss_target: Ord.factor w/ 3 levels "Vanishing"<"Mislabeling"<..: 1 2 3 1 2 3 1 2 3 1 ...
##   ..$ data       : list<tibble[,15]> [1:15]
##   ..$ mod        :List of 15
##   ..- attr(*, "groups")= tibble [15 x 3] (S3: tbl_df/tbl/data.frame)
##  $ tidied: gropd_df [45 x 20] (S3: grouped_df/tbl_df/tbl/data.frame)
##   ..$ model_name    : Ord.factor w/ 5 levels "YOLOv3"<"SSD"<..: 1 1 1 1 1 1 1 1 1 2 ...
##   ..$ loss_target   : Ord.factor w/ 3 levels "Vanishing"<"Mislabeling"<..: 1 1 1 2 2 2 3 3 3 1 ...
##   ..$ term          : chr [1:45] "bbox_dist" "bbox_size_perturb" "bbox_dist:bbox_size_perturb" "bbox_
##   ..$ variable      : chr [1:45] "bbox_dist" "bbox_size_perturb" "bbox_dist:bbox_size_perturb" "bbox_
##   ..$ var_label     : Named chr [1:45] "bbox_dist" "bbox_size_perturb" "bbox_dist * bbox_size_perturl
##   .. ..- attr(*, "names")= chr [1:45] "bbox_dist" "bbox_size_perturb" "bbox_dist:bbox_size_perturb"
##   ..$ var_class     : Named chr [1:45] "numeric" "numeric" NA "numeric" ...
##   .. ..- attr(*, "names")= chr [1:45] "bbox_dist" "bbox_size_perturb" "" "bbox_dist" ...
##   ..$ var_type      : chr [1:45] "continuous" "continuous" "interaction" "continuous" ...
##   ..$ var_nlevels   : int [1:45] NA NA NA NA NA NA NA NA NA NA ...
##   ..$ contrasts     : chr [1:45] NA NA NA NA ...
##   ..$ contrasts_type: chr [1:45] NA NA NA NA ...
##   ..$ reference_row : logi [1:45] NA NA NA NA NA NA ...
##   ..$ label         : Named chr [1:45] "bbox_dist" "bbox_size_perturb" "bbox_dist * bbox_size_perturl
##   .. ..- attr(*, "names")= chr [1:45] "bbox_dist" "bbox_size_perturb" "bbox_dist:bbox_size_perturb"
##   ..$ n_obs         : Named num [1:45] 4000 4000 4000 4000 4000 4000 4000 4000 4000 4000 ...
##   .. ..- attr(*, "names")= chr [1:45] "bbox_dist" "bbox_size_perturb" "bbox_dist:bbox_size_perturb"
##   ..$ n_event       : Named num [1:45] 1062 1062 1062 934 934 ...
##   .. ..- attr(*, "names")= chr [1:45] "bbox_dist" "bbox_size_perturb" "bbox_dist:bbox_size_perturb"
##   ..$ estimate      : num [1:45] -8.54 26.83 -79.93 -8.47 10.99 ...
##   ..$ std.error     : num [1:45] 0.694 1.719 8.924 0.615 0.956 ...
##   ..$ statistic     : num [1:45] -12.29 15.61 -8.96 -13.78 11.5 ...
##   ..$ p.value       : num [1:45] 1.01e-34 6.26e-55 3.34e-19 3.45e-43 1.32e-30 ...
##   ..$ conf.low      : num [1:45] -9.93 23.55 -97.84 -9.71 9.17 ...
##   ..$ conf.high     : num [1:45] -7.21 30.29 -62.85 -7.3 12.92 ...
##   ..- attr(*, "groups")= tibble [15 x 3] (S3: tbl_df/tbl/data.frame)
##   .. ..- attr(*, ".drop")= logi TRUE
```

```
reg_est <- reg_res$tidied

ext_sig(reg_est, "neg", "bbox_dist")
```

```
## ----------bbox_dist----------
## Total 15 predictors:
## 15 (100%) significant;
## 15 (100%) neg
```

```
## # A tibble: 15 x 9
## # Groups:   model_name, loss_target [15]
##    model_name    loss_target term  estimate std.error statistic p.value conf.low
##    <ord>         <ord>       <chr>    <dbl>     <dbl>     <dbl>   <dbl>    <dbl>
##  1 YOLOv3        Vanishing   bbox~    -8.54     0.694    -12.3        0    -9.93
##  2 YOLOv3        Mislabeling bbox~    -8.47     0.615    -13.8        0    -9.71
##  3 YOLOv3        Untargeted  bbox~   -15.9      1.37     -11.6        0   -18.6
##  4 SSD           Vanishing   bbox~   -18.4      1.16     -15.9        0   -20.8
##  5 SSD           Mislabeling bbox~   -19.7      1.31     -15.0        0   -22.3
##  6 SSD           Untargeted  bbox~   -21.7      1.54     -14.1        0   -24.9
##  7 RetinaNet     Vanishing   bbox~   -35.3      3.25     -10.9        0   -41.9
##  8 RetinaNet     Mislabeling bbox~   -49.8      6.49      -7.68       0   -63.3
##  9 RetinaNet     Untargeted  bbox~   -13.9      1.41      -9.84       0   -16.8
## 10 Faster R-CNN  Vanishing   bbox~   -21.0      3.20      -6.56       0   -27.7
## 11 Faster R-CNN  Mislabeling bbox~   -17.8      3.24      -5.51       0   -24.7
## 12 Faster R-CNN  Untargeted  bbox~   -19.1      1.79     -10.7        0   -22.7
## 13 Cascade R-CNN Vanishing   bbox~   -32.5      4.07      -7.99       0   -41.0
## 14 Cascade R-CNN Mislabeling bbox~   -27.7      4.73      -5.86       0   -37.8
## 15 Cascade R-CNN Untargeted  bbox~   -22.5      2.47      -9.12       0   -27.6
## # i 1 more variable: conf.high <dbl>
```

```
ext_sig(reg_est, "pos", "bbox_size_perturb")
```

```
## ----------bbox_size_perturb----------
## Total 15 predictors:
## 11 (73%) significant;
## 11 (73%) pos
```

```
## # A tibble: 11 x 9
## # Groups:   model_name, loss_target [11]
##    model_name    loss_target term  estimate std.error statistic p.value conf.low
##    <ord>         <ord>       <chr>    <dbl>     <dbl>     <dbl>   <dbl>    <dbl>
##  1 YOLOv3        Vanishing   bbox~    26.8      1.72      15.6    0       23.6
##  2 YOLOv3        Mislabeling bbox~    11.0      0.956     11.5    0        9.17
##  3 SSD           Vanishing   bbox~     7.27     0.813      8.95   0        5.73
##  4 SSD           Mislabeling bbox~     3.38     0.612      5.53   0        2.22
##  5 SSD           Untargeted  bbox~     1.39     0.545      2.55   0.011    0.336
##  6 RetinaNet     Vanishing   bbox~     2.32     0.695      3.33   0.001    0.993
##  7 RetinaNet     Untargeted  bbox~     2.99     0.539      5.54   0        1.94
##  8 Faster R-CNN  Vanishing   bbox~     6.10     1.23       4.96   0        3.75
##  9 Cascade R-CNN Vanishing   bbox~     7.51     0.966      7.78   0        5.71
## 10 Cascade R-CNN Mislabeling bbox~     4.90     0.797      6.15   0        3.35
## 11 Cascade R-CNN Untargeted  bbox~     2.11     0.648      3.26   0.001    0.833
## # i 1 more variable: conf.high <dbl>
```

```
ext_sig(reg_est, "both", "bbox_dist:bbox_size_perturb")
```

```
## ----------bbox_dist:bbox_size_perturb----------
## Total 15 predictors:
## 10 (67%) significant;
## 10 (67%) both
```

```
## # A tibble: 10 x 9
## # Groups:   model_name, loss_target [10]
##    model_name    loss_target term  estimate std.error statistic p.value conf.low
##    <ord>         <ord>       <chr>    <dbl>     <dbl>     <dbl>   <dbl>    <dbl>
##  1 YOLOv3        Vanishing   bbox~    -79.9      8.92     -8.96   0        -97.8
##  2 YOLOv3        Mislabeling bbox~    -24.1      5.92     -4.08   0        -36.0
##  3 YOLOv3        Untargeted  bbox~     39.5      6.52      6.06   0         26.7
##  4 SSD           Mislabeling bbox~     24.0      6.04      3.97   0         12.0
##  5 SSD           Untargeted  bbox~     34.2      6.42      5.32   0         21.4
##  6 RetinaNet     Vanishing   bbox~     47.0     11.2       4.19   0         24.3
##  7 RetinaNet     Untargeted  bbox~     28.1      5.11      5.49   0         18.1
##  8 Faster R-CNN  Vanishing   bbox~    -83.5     28.5      -2.93   0.003   -144.
##  9 Faster R-CNN  Untargeted  bbox~     61.5      6.97      8.82   0         48.4
## 10 Cascade R-CNN Vanishing   bbox~   -106.      31.1      -3.42   0.001   -172.
## # i 1 more variable: conf.high <dbl>
```

```
print_statistics(reg_est, table_caption(pred_name, main_pt))
```

Table 5: We run a logistic model regressing success against perturb-target distance (relative to image width/height) and perturb box size (relative to image width/height) in the randomized attack experiment. Larger perturb objects significantly increase success rates for all models and attacks, except for mislabeling attack on Faster R-CNN, after controlling for perturb-target distances. Shorter perturb-target distances significantly increase success rates for all models and attacks, after controlling for perturb object sizes. Table headers are explained in Appendix **??**.

Group	Regression							
Attack	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
YOLOv3								
Vanishing	distance	*	-8.536	0.694	-12.292	0.000	-9.929	-7.207
	size	*	26.831	1.719	15.610	0.000	23.555	30.294
	distance * size	*	-79.933	8.924	-8.957	0.000	-97.839	-62.847

Mislabeling	distance	*	-8.473	0.615	-13.778	0.000	-9.707	-7.297
	size	*	10.991	0.956	11.500	0.000	9.169	12.915
	distance * size	*	-24.117	5.917	-4.076	0.000	-35.972	-12.770
Untargeted	distance	*	-15.869	1.366	-11.614	0.000	-18.640	-13.284
	size		0.308	0.704	0.437	0.662	-1.087	1.678
	distance * size	*	39.532	6.522	6.061	0.000	26.743	52.347
SSD								
Vanishing	distance	*	-18.433	1.159	-15.903	0.000	-20.766	-16.222
	size	*	7.274	0.813	8.948	0.000	5.728	8.915
	distance * size		7.663	6.391	1.199	0.231	-5.139	19.931
Mislabeling	distance	*	-19.702	1.311	-15.023	0.000	-22.349	-17.208
	size	*	3.384	0.612	5.531	0.000	2.217	4.617
	distance * size	*	23.987	6.040	3.971	0.000	11.954	35.660
Untargeted	distance	*	-21.725	1.544	-14.069	0.000	-24.852	-18.799
	size	*	1.389	0.545	2.547	0.011	0.336	2.478
	distance * size	*	34.171	6.423	5.320	0.000	21.425	46.643
RetinaNet								
Vanishing	distance	*	-35.303	3.249	-10.864	0.000	-41.932	-29.191
	size	*	2.317	0.695	3.334	0.001	0.993	3.717
	distance * size	*	46.975	11.215	4.189	0.000	24.285	68.263
Mislabeling	distance	*	-49.847	6.486	-7.685	0.000	-63.277	-37.849
	size		1.056	1.187	0.889	0.374	-1.244	3.427
	distance * size		37.912	25.512	1.486	0.137	-15.784	84.709
Untargeted	distance	*	-13.895	1.412	-9.843	0.000	-16.788	-11.254
	size	*	2.989	0.539	5.544	0.000	1.938	4.054
	distance * size	*	28.072	5.111	5.493	0.000	18.127	38.241
Faster R-CNN								
Vanishing	distance	*	-21.030	3.204	-6.564	0.000	-27.739	-15.185
	size	*	6.096	1.228	4.962	0.000	3.747	8.571
	distance * size	*	-83.474	28.510	-2.928	0.003	-144.255	-31.915
Mislabeling	distance	*	-17.846	3.240	-5.507	0.000	-24.720	-12.034
	size		1.205	1.719	0.701	0.483	-2.408	4.397
	distance * size		-54.135	39.695	-1.364	0.173	-142.163	14.635
Untargeted	distance	*	-19.078	1.789	-10.665	0.000	-22.746	-15.729
	size		-0.274	0.719	-0.381	0.703	-1.711	1.113
	distance * size	*	61.468	6.966	8.824	0.000	48.369	75.700
Cascade R-CNN								
Vanishing	distance	*	-32.490	4.066	-7.991	0.000	-40.976	-25.029
	size	*	7.513	0.966	7.779	0.000	5.711	9.508
	distance * size	*	-106.218	31.092	-3.416	0.001	-172.083	-49.911
Mislabeling	distance	*	-27.708	4.732	-5.856	0.000	-37.836	-19.260
	size	*	4.898	0.797	6.146	0.000	3.354	6.485

	distance * size		-49.344	27.328	-1.806	0.071	-107.414	-0.192
Untargeted	distance	*	-22.497	2.467	-9.120	0.000	-27.587	-17.915
	size	*	2.113	0.648	3.258	0.001	0.833	3.381
	distance * size		5.873	11.482	0.512	0.609	-18.022	27.276

```r
reg_mod <- reg_res$mod

newdata <- expand_grid(
  bbox_dist = linear_space(data$bbox_dist),
  bbox_size_perturb = linear_space(data$bbox_size_perturb)
) |>
  glimpse()
```

```
## Rows: 10,000
## Columns: 2
## $ bbox_dist         <dbl> 1.180172e-05, 1.180172e-05, 1.180172e-05, 1.180172e-~
## $ bbox_size_perturb <dbl> 2.671581e-05, 8.839428e-03, 1.765214e-02, 2.646485e-~
```

```r
reg_pred <- reg_mod |>
  summarize(augment(mod, newdata = newdata, type.predict = "response")) |>
  rename(success = .fitted) |>
  glimpse()
```

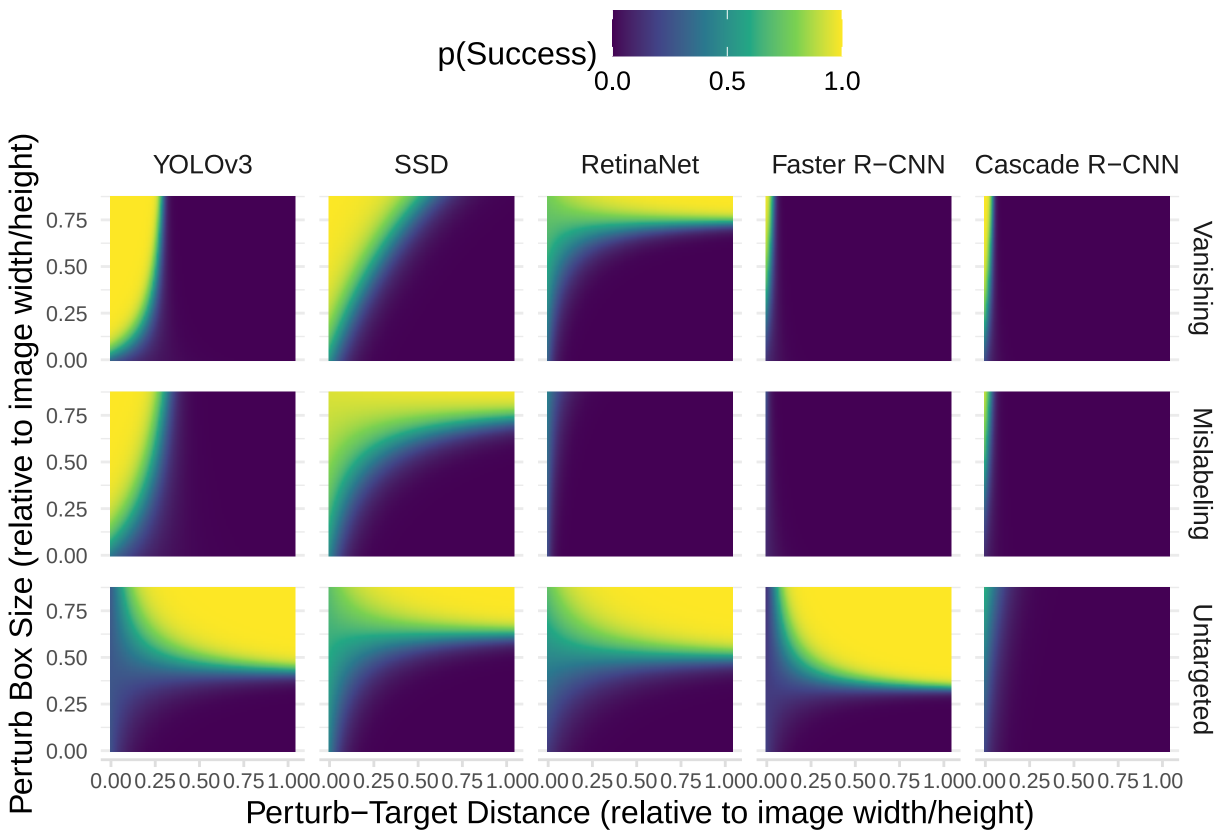
```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## `summarise()` has grouped output by 'model_name', 'loss_target'. You can
## override using the `.groups` argument.

## Rows: 150,000
## Columns: 5
## Groups: model_name, loss_target [15]
## $ model_name        <ord> YOLOv3, YOLOv3, YOLOv3, YOLOv3, YOLOv3, YOLOv3, YOLO~
## $ loss_target       <ord> Vanishing, Vanishing, Vanishing, Vanishing, Vanishin~
## $ bbox_dist         <dbl> 1.180172e-05, 1.180172e-05, 1.180172e-05, 1.180172e-~
## $ bbox_size_perturb <dbl> 2.671581e-05, 8.839428e-03, 1.765214e-02, 2.646485e-~
## $ success           <dbl> 0.3376685, 0.3923941, 0.4499635, 0.5089050, 0.567600~
```

```r
g <- reg_pred |> ggplot(aes(bbox_dist, bbox_size_perturb, fill = success)) +
  geom_raster(interpolate = TRUE)

graph_dist_size(g)
```



```r
# get success rate on ground truth sampled images
gt_success_data <- bbox_raw_data |>
  filter(bbox_type == "ground_truth") |>
  # loss_target is not relevant
  count(model_name, bbox_class, bbox_res_eval) |>
  # get success probability
  # https://stackoverflow.com/a/37448040/19655086
  as_tibble() |>
  pivot_wider(names_from = "bbox_res_eval", values_from = n) |>
  # not every class has tp and fn
  replace_na(list(tp = 0, fn = 0)) |>
  mutate(gt_p_success = tp / (fn + tp)) |>
  # some 0/0
  drop_na(gt_p_success) |>
  select(model_name, bbox_class, gt_p_success) |>
  glimpse()
```

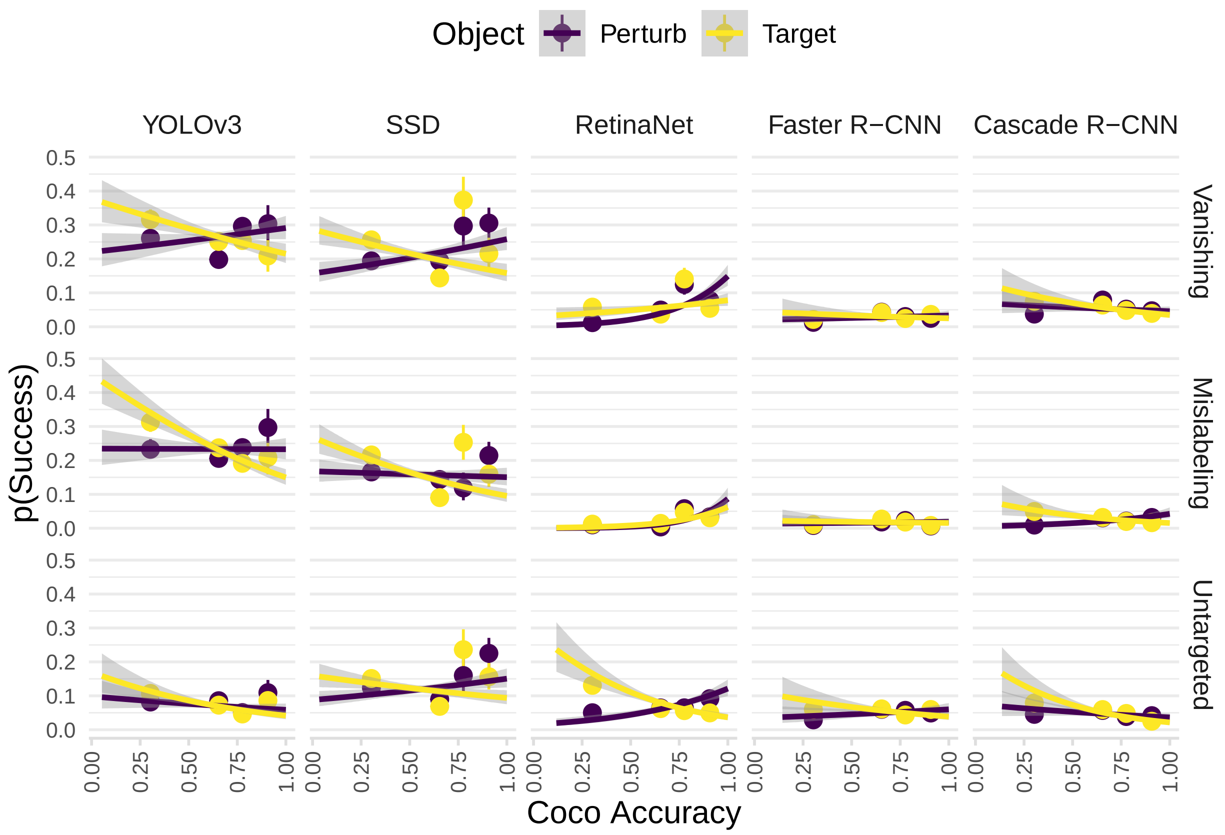
```
## Rows: 378
## Columns: 3
## $ model_name   <ord> YOLOv3, YOLOv3, YOLOv3, YOLOv3, YOLOv3, YOLOv3, YOLOv3, Y~
## $ bbox_class   <chr> "airplane", "apple", "backpack", "banana", "baseball bat"~
## $ gt_p_success <dbl> 1.0000000, 0.7196110, 0.2938005, 0.1124452, 0.5025961, 0.~
```

```r
# by model_name, bbox_class
# some classes are not evaluated
gt_success_data <- bbox_conf_data |>
  inner_join(gt_success_data) |>
  glimpse()
```

```
## Joining with `by = join_by(bbox_class, model_name)`
```

```
## Rows: 120,000
## Columns: 43
## $ fname                       <chr> "/Users/zbli/Documents/Documents - ZhaoBin'~
## $ sample_id                   <chr> "65ed3a88141a475067f32706", "65ed3a88141a47~
## $ sample_path                 <chr> "/projects/f_ps848_1/zhaobin/adversarial/co~
## $ sample_width                <int> 640, 640, 500, 640, 480, 640, 640, 640, 640~
## $ sample_height               <int> 480, 427, 332, 425, 640, 480, 480, 480, 640~
## $ sample_mislabel_class       <chr> "horse", "truck", "surfboard", "horse", "ca~
## $ sample_mislabel_proba       <dbl> 6.615031e-05, 4.219168e-02, 4.392489e-05, 1~
## $ sample_attack               <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, T~
## $ sample_vanish               <lgl> FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, FA~
## $ sample_mislabel_intended    <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ sample_success              <lgl> FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, FA~
## $ sample_mislabel             <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ bbox_id                     <chr> "65ed3aa3141a475067f3ca3e", "65ed3aa3141a47~
## $ bbox_class                  <chr> "clock", "car", "person", "person", "donut"~
## $ bbox_xywhn                  <list<double>> <0.32723613, 0.26601949, 0.0435188~
## $ bbox_conf                   <dbl> 0.9305881, 0.3433506, 0.9882318, 0.9988949,~
## $ bbox_res_eval               <chr> "tp", "tp", "tp", "tp", "tp", "tp", "tp", "~
## $ bbox_iou_eval               <dbl> 0.8860679, 0.7609860, 0.9454082, 0.9299325,~
## $ bbox_res_pgd_eval           <chr> "tp", "tp", "fn", "tp", "tp", "tp", "tp", "~
## $ bbox_iou_pgd_eval           <dbl> 1.0000000, 1.0000000, NA, 0.9999969, 1.0000~
## $ bbox_res_pgd_mislabel_eval  <chr> NA, NA, "fn", NA, NA, NA, NA, NA, NA, NA, N~
## $ bbox_iou_pgd_mislabel_eval  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ bbox_type                   <chr> "predictions", "predictions", "predictions"~
## $ bbox_mislabel               <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ num_iteration               <dbl> 200, 200, 200, 200, 200, 200, 200, 200, 200~
## $ max_norm                    <dbl> 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0~
## $ model_name                  <ord> Cascade R-CNN, Cascade R-CNN, Cascade R-CNN~
## $ loss_target                 <ord> Mislabeling, Mislabeling, Mislabeling, Misl~
## $ attack_bbox                 <chr> "predictions", "predictions", "predictions"~
## $ perturb_fun                 <chr> "perturb_inside", "perturb_inside", "pertur~
## $ sample_count                <dbl> 247, 247, 247, 247, 247, 247, 247, 247, 247~
## $ attack_count                <dbl> 200, 200, 200, 200, 200, 200, 200, 200, 200~
## $ success_count               <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8~
## $ vanish_count                <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ mislabel_count              <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6~
## $ mislabel_intended_count     <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6~
## $ target_max_conf             <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ perturb_min_size            <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ bbox_max_dist               <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ target_or_perturb           <ord> Target, Target, Target, Target, Target, Tar~
## $ target_or_perturb_boolean   <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, T~
## $ success                     <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ gt_p_success                <dbl> 0.9090909, 0.7741935, 0.8199719, 0.8199719,~
```

```r
gt_success_data |>
  graph_attr(gt_p_success, "COCO Accuracy")
```



```
pred_name <- "mean COCO accuracy for the target class"
main_pt <- "the results are mixed after controlling for target class confidence"

cap <- graph_caption(pred_name, glue("Although higher {pred_name} seem to decrease success rates, {main_

gt_success_graph <- gt_success_data |> filter(target_or_perturb == "Target")
gt_success_graph |>
  graph_attr(gt_p_success, pred_name)
```

```
model <- partial(glm_model, predictor = "gt_p_success * bbox_conf")
data <- gt_success_graph

reg_est <- get_tidied_reg(model, data)
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## `summarise()` has grouped output by 'model_name', 'loss_target'. You can
## override using the `.groups` argument.
```

```
# there are both significantly positive and negative gt_p_success,
# and the interaction term is relatively large
ext_sig(reg_est, "neg", "gt_p_success")
```

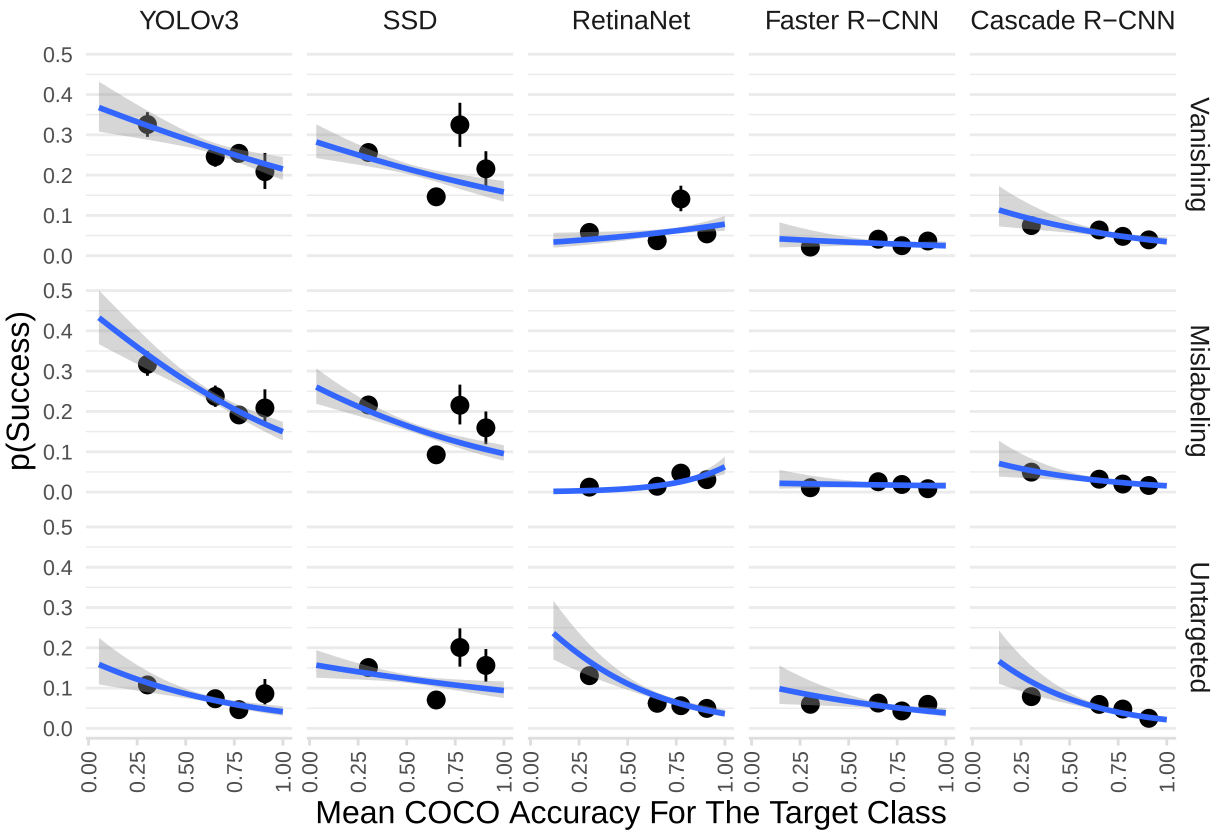


Figure 4: Although higher mean COCO accuracy for the target class seem to decrease success rates, the results are mixed after controlling for target class confidence (Table 6) even with 0.05 max-norm: The binned summaries and regression trendlines graph success proportion against mean COCO accuracy for the target class in the randomized attack experiment. Bins are split into quantiles. Errors are 95% confidence intervals

```
## ----------gt_p_success----------
## Total 15 predictors:
## 8 (53%) significant;
## 2 (13%) neg
```

```
## # A tibble: 2 x 9
## # Groups:   model_name, loss_target [2]
##   model_name    loss_target term   estimate std.error statistic p.value conf.low
##   <ord>         <ord>       <chr>     <dbl>     <dbl>     <dbl>   <dbl>    <dbl>
## 1 Cascade R-CNN Vanishing   gt_p_~    -4.25      1.49     -2.85   0.004    -7.16
## 2 Cascade R-CNN Mislabeling gt_p_~    -4.57      1.81     -2.53   0.011    -8.08
## # i 1 more variable: conf.high <dbl>
```

```
ext_sig(reg_est, "pos", "gt_p_success")
```

```
## ----------gt_p_success----------
## Total 15 predictors:
## 8 (53%) significant;
## 6 (40%) pos
```

```
## # A tibble: 6 x 9
## # Groups:   model_name, loss_target [6]
##   model_name loss_target term      estimate std.error statistic p.value conf.low
##   <ord>      <ord>       <chr>        <dbl>     <dbl>     <dbl>  <dbl>    <dbl>
## 1 SSD        Vanishing   gt_p_suc~     3.77     0.582      6.48  0        2.64
## 2 SSD        Mislabeling gt_p_suc~     4.38     0.63       6.95  0        3.15
## 3 SSD        Untargeted  gt_p_suc~     3.38     0.681      4.96  0        2.05
## 4 RetinaNet  Vanishing   gt_p_suc~     3.27     1.39       2.35  0.019    0.576
## 5 RetinaNet  Mislabeling gt_p_suc~    11.0      2.73       4.02  0        5.68
## 6 RetinaNet  Untargeted  gt_p_suc~     3.55     1.29       2.75  0.006    1.03
## # i 1 more variable: conf.high <dbl>
```

```r
ext_sig(reg_est, "both", "gt_p_success:bbox_conf")
```

```
## ----------gt_p_success:bbox_conf----------
## Total 15 predictors:
## 10 (67%) significant;
## 10 (67%) both

## # A tibble: 10 x 9
## # Groups:   model_name, loss_target [10]
##    model_name    loss_target term  estimate std.error statistic p.value conf.low
##    <ord>         <ord>       <chr>    <dbl>     <dbl>     <dbl>   <dbl>    <dbl>
##  1 YOLOv3        Vanishing   gt_p~    -2.05     1.01      -2.03   0.042    -4.03
##  2 YOLOv3        Mislabeling gt_p~    -3.48     1.06      -3.27   0.001    -5.57
##  3 YOLOv3        Untargeted  gt_p~    -4.86     1.91      -2.54   0.011    -8.61
##  4 SSD           Vanishing   gt_p~    -6.66     0.854     -7.79   0        -8.34
##  5 SSD           Mislabeling gt_p~    -8.65     0.976     -8.86   0       -10.6
##  6 SSD           Untargeted  gt_p~    -6.06     1.11      -5.48   0        -8.24
##  7 RetinaNet     Mislabeling gt_p~   -11.7      5.71      -2.05   0.04    -22.6
##  8 RetinaNet     Untargeted  gt_p~    -9.35     2.76      -3.39   0.001   -14.8
##  9 Cascade R-CNN Vanishing   gt_p~     4.33     1.96       2.21   0.027     0.483
## 10 Cascade R-CNN Mislabeling gt_p~     5.32     2.64       2.02   0.044     0.152
## # i 1 more variable: conf.high <dbl>
```

```r
print_statistics(reg_est, table_caption(
  glue("{pred_name}, with target confidence as covariate,"),
  glue("{main_pt} and the relatively large interaction terms make interpretation challenging")
))
```

Table 6: We run a logistic model regressing success against mean COCO accuracy for the target class, with target confidence as covariate, in the randomized attack experiment. The results are mixed after controlling for target class confidence and the relatively large interaction terms make interpretation challenging. Table headers are explained in Appendix ??.

Group	Regression							
Attack	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
YOLOv3								
Vanishing	accuracy		0.842	0.747	1.127	0.260	-0.619	2.313
	confidence		0.368	0.671	0.548	0.584	-0.945	1.688
	accuracy * confidence	*	-2.046	1.007	-2.031	0.042	-4.026	-0.076
Mislabeling	accuracy		1.231	0.754	1.631	0.103	-0.247	2.712
	confidence		-0.139	0.700	-0.198	0.843	-1.514	1.234
	accuracy * confidence	*	-3.481	1.065	-3.270	0.001	-5.571	-1.396
Untargeted	accuracy		1.941	1.117	1.737	0.082	-0.240	4.143

	confidence		-1.715	1.230	-1.394	0.163	-4.155	0.671
	accuracy * confidence	*	-4.861	1.913	-2.541	0.011	-8.612	-1.112
SSD								
Vanishing	accuracy	*	3.774	0.582	6.485	0.000	2.640	4.923
	confidence	*	2.184	0.491	4.451	0.000	1.226	3.150
	accuracy * confidence	*	-6.655	0.854	-7.789	0.000	-8.340	-4.990
Mislabeling	accuracy	*	4.376	0.630	6.950	0.000	3.148	5.618
	confidence	*	2.449	0.538	4.550	0.000	1.395	3.506
	accuracy * confidence	*	-8.650	0.976	-8.864	0.000	-10.573	-6.746
Untargeted	accuracy	*	3.376	0.681	4.955	0.000	2.047	4.720
	confidence		0.423	0.626	0.677	0.499	-0.809	1.646
	accuracy * confidence	*	-6.063	1.106	-5.480	0.000	-8.239	-3.902
RetinaNet								
Vanishing	accuracy	*	3.267	1.389	2.353	0.019	0.576	6.018
	confidence		-0.776	2.077	-0.374	0.709	-4.879	3.260
	accuracy * confidence		-2.512	2.651	-0.948	0.343	-7.702	2.686
Mislabeling	accuracy	*	10.978	2.731	4.020	0.000	5.683	16.358
	confidence		3.473	4.602	0.755	0.450	-5.826	12.146
	accuracy * confidence	*	-11.692	5.707	-2.049	0.040	-22.608	-0.344
Untargeted	accuracy	*	3.553	1.292	2.751	0.006	1.029	6.093
	confidence		0.863	1.920	0.449	0.653	-2.964	4.566
	accuracy * confidence	*	-9.351	2.760	-3.388	0.001	-14.760	-3.935
Faster R-CNN								
Vanishing	accuracy		-1.752	1.802	-0.973	0.331	-5.202	1.874
	confidence	*	-6.201	2.110	-2.939	0.003	-10.372	-2.093
	accuracy * confidence		3.626	2.762	1.313	0.189	-1.797	9.030
Mislabeling	accuracy		2.740	2.469	1.110	0.267	-1.989	7.689
	confidence		-3.313	3.126	-1.060	0.289	-9.642	2.613
	accuracy * confidence		-2.724	4.126	-0.660	0.509	-10.668	5.473
Untargeted	accuracy		1.841	1.415	1.301	0.193	-0.897	4.655
	confidence		-2.543	1.607	-1.583	0.114	-5.733	0.572
	accuracy * confidence		-2.728	2.162	-1.262	0.207	-6.949	1.529
Cascade R-CNN								
Vanishing	accuracy	*	-4.247	1.491	-2.848	0.004	-7.156	-1.298
	confidence	*	-4.563	1.413	-3.229	0.001	-7.328	-1.779
	accuracy * confidence	*	4.330	1.956	2.214	0.027	0.483	8.158
Mislabeling	accuracy	*	-4.568	1.806	-2.530	0.011	-8.081	-0.985
	confidence	*	-6.823	1.939	-3.519	0.000	-10.663	-3.046
	accuracy * confidence	*	5.322	2.638	2.017	0.044	0.152	10.503
Untargeted	accuracy		-0.017	1.423	-0.012	0.990	-2.791	2.794
	confidence		-1.750	1.449	-1.207	0.227	-4.607	1.083

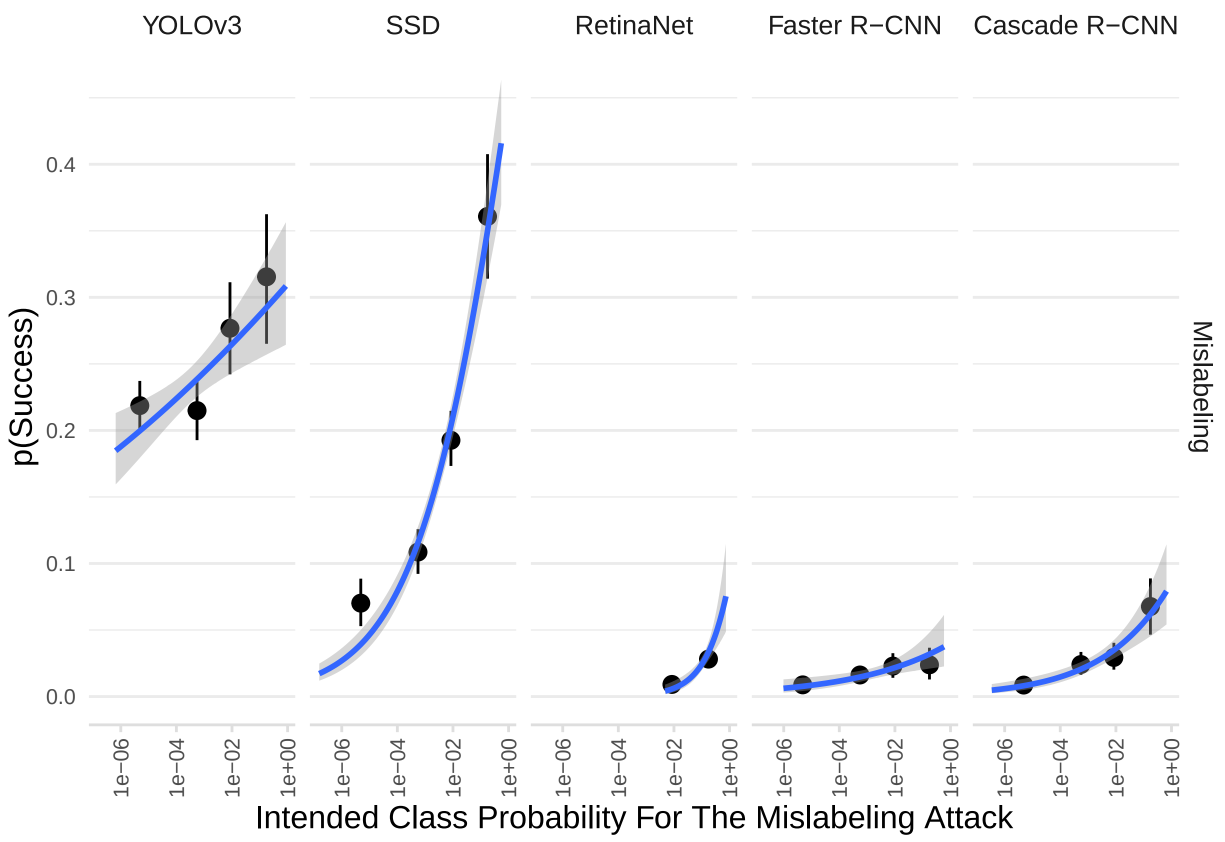


Figure 5: Although intended class probability seem to increase success rates for the mislabeling attack, it does not predict success rates after controlling for target class confidence, except for RetinaNet (Table 7) even with 0.05 max-norm: The binned summaries and regression trendlines graph success proportion against intended class probability in the randomized attack experiment. Bins are split into quantiles. Errors are 95% confidence intervals

accuracy * confidence	-2.732	2.037	-1.341	0.180	-6.726	1.265

```r
# restrict to mislabeling
bbox_proba_graph <- bbox_conf_data |>
  filter(loss_target == "Mislabeling" & target_or_perturb == "Target")

# check is not logit
stopifnot(max(bbox_proba_graph$sample_mislabel_proba) <= 1 && min(bbox_proba_graph$sample_mislabel_proba
```

```r
pred_name <- "intended class probability"
att_name <- "for the mislabeling attack"

main_pt <- glue("does not predict success rates after controlling for target class confidence, except fo
cap <- graph_caption(pred_name, glue("Although {pred_name} seem to increase success rates {att_name}, it

g <- bbox_proba_graph |>
  graph_attr(sample_mislabel_proba, glue("{pred_name} {att_name}"), scale_x_log10())
```

```r
model <- partial(glm_model, predictor = "log(sample_mislabel_proba) * bbox_conf")
data <- bbox_proba_graph
```

```
reg_est <- get_tidied_reg(model, data)
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## `summarise()` has grouped output by 'model_name', 'loss_target'. You can
## override using the `.groups` argument.
```

```
ext_sig(reg_est, "pos", "log(sample_mislabel_proba)")
```

```
## ----------log(sample_mislabel_proba)----------
## Total 5 predictors:
## 3 (60%) significant;
## 2 (40%) pos

## # A tibble: 2 x 9
## # Groups:   model_name, loss_target [2]
##   model_name loss_target term      estimate std.error statistic p.value conf.low
##   <ord>      <ord>       <chr>        <dbl>     <dbl>     <dbl>   <dbl>    <dbl>
## 1 SSD        Mislabeling log(samp~    0.196     0.055      3.57   0        0.089
## 2 RetinaNet  Mislabeling log(samp~    1.12      0.373      2.99   0.003    0.374
## # i 1 more variable: conf.high <dbl>
```

```
ext_sig(reg_est, "both", "log(sample_mislabel_proba):bbox_conf")
```

```
## ----------log(sample_mislabel_proba):bbox_conf----------
## Total 5 predictors:
## 1 (20%) significant;
## 1 (20%) both

## # A tibble: 1 x 9
## # Groups:   model_name, loss_target [1]
##   model_name loss_target term      estimate std.error statistic p.value conf.low
##   <ord>      <ord>       <chr>        <dbl>     <dbl>     <dbl>   <dbl>    <dbl>
## 1 YOLOv3     Mislabeling log(samp~    0.317     0.062      5.14       0    0.196
## # i 1 more variable: conf.high <dbl>
```

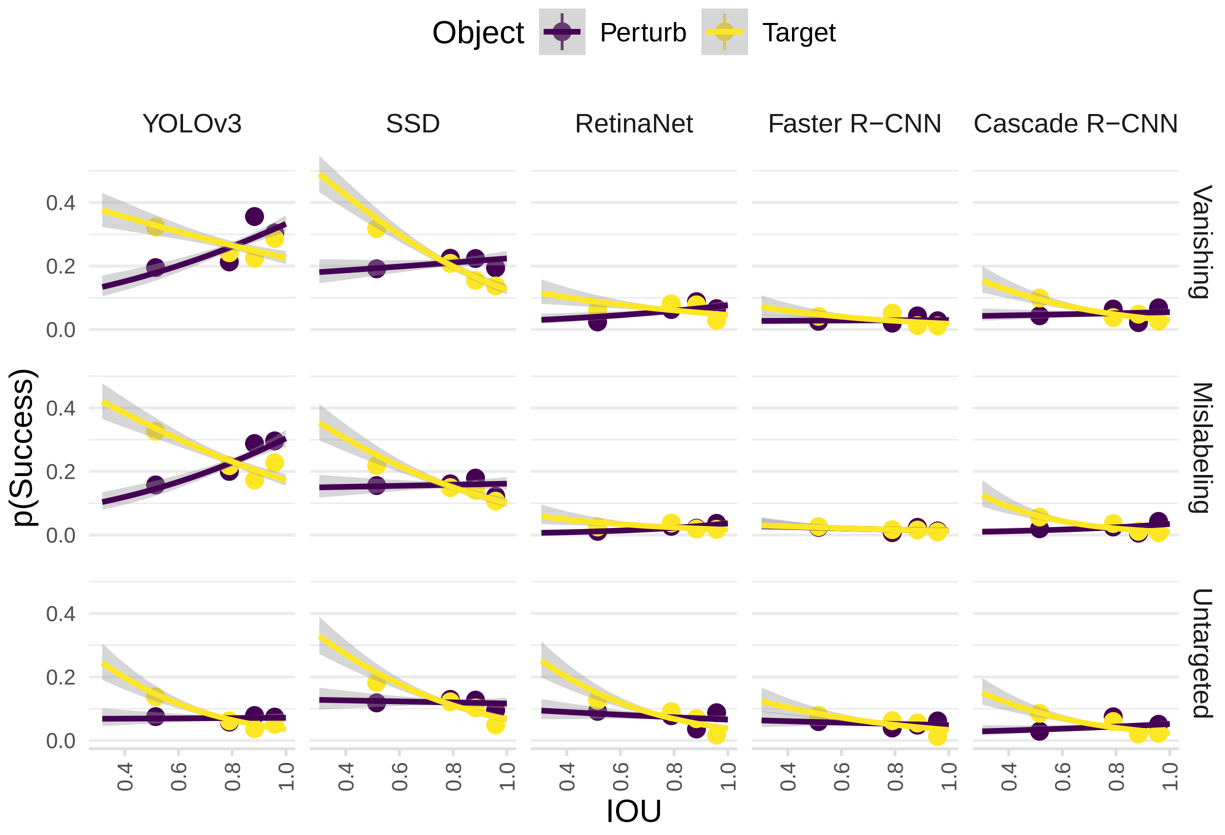
```
print_statistics(reg_est, table_caption(glue("log({pred_name}) {att_name}, with predicted class's confi
```

Table 7: We run a logistic model regressing success against log(intended class probability) for the mislabeling attack, with predicted class's confidence as covariate, in the randomized attack experiment. Intended class probability does not predict success rates after controlling for target class confidence, except for RetinaNet. Table headers are explained in Appendix ??.

Group	Regression							
Model	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
Mislabeling								
YOLOv3	log(probability)	*	-0.183	0.042	-4.344	0.000	-0.266	-0.101
	confidence		0.119	0.522	0.227	0.820	-0.904	1.143

	log(probability) * confidence	*	0.317	0.062	5.140	0.000	0.196	0.438
SSD	log(probability)	*	0.196	0.055	3.574	0.000	0.089	0.304
	confidence	*	-1.546	0.503	-3.071	0.002	-2.532	-0.558
	log(probability) * confidence		0.011	0.078	0.146	0.884	-0.141	0.166
RetinaNet	log(probability)	*	1.117	0.373	2.993	0.003	0.374	1.837
	confidence	*	-8.002	1.997	-4.006	0.000	-11.970	-4.136
	log(probability) * confidence		-1.384	0.757	-1.828	0.067	-2.822	0.145
Faster R-CNN	log(probability)		0.158	0.120	1.314	0.189	-0.080	0.393
	confidence	*	-7.667	1.544	-4.964	0.000	-10.765	-4.692
	log(probability) * confidence		-0.330	0.196	-1.684	0.092	-0.709	0.061
Cascade R-CNN	log(probability)		0.096	0.111	0.864	0.388	-0.123	0.313
	confidence	*	-2.499	1.024	-2.440	0.015	-4.493	-0.470
	log(probability) * confidence		0.020	0.153	0.133	0.894	-0.275	0.326

```r
# bbox iou always based on predictions bbox like confidence
bbox_conf_data |>
  graph_attr(bbox_iou_eval, " IOU ")
```



```r
# restrict to target bbox and untargeted attack only
pred_name <- "target iou for the untargeted attack"
main_pt <- glue("{pred_name} increases success rates on all models")

cap <- graph_caption(pred_name, main_pt, params$norm)
```

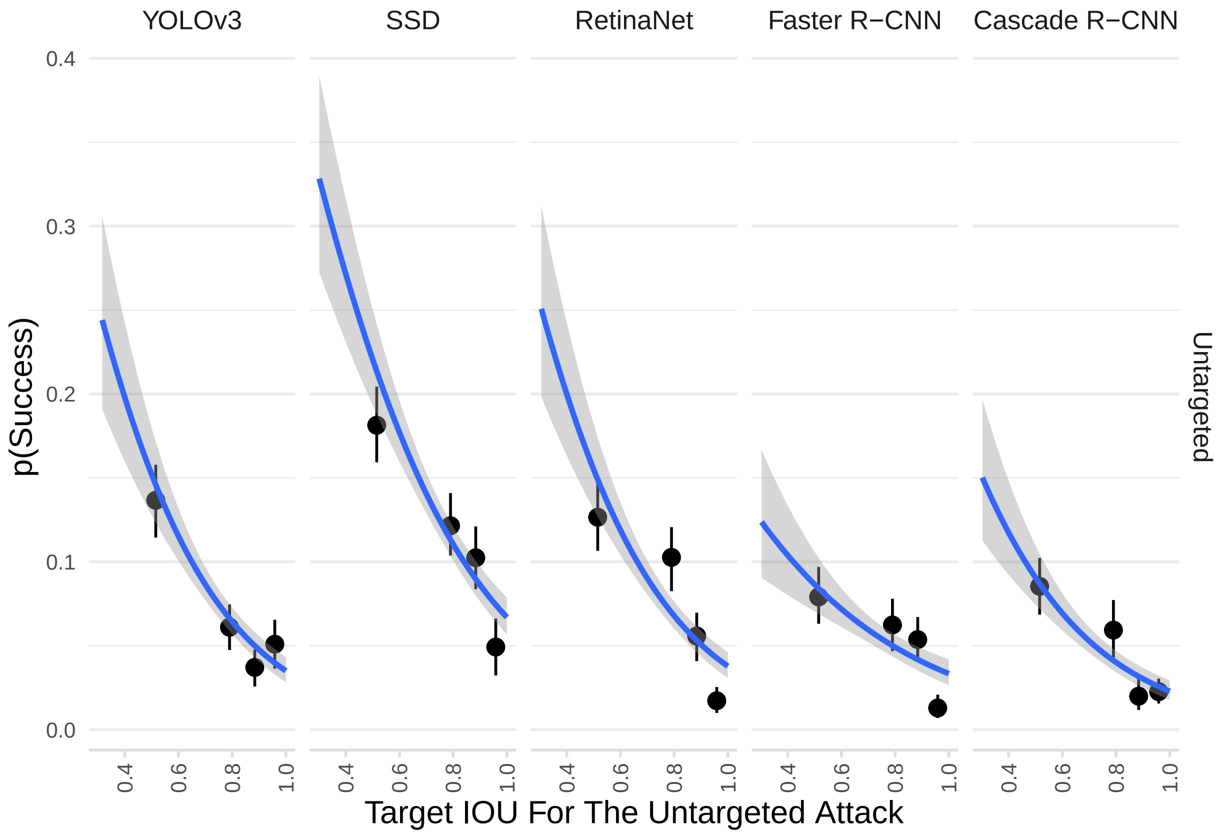


Figure 6: Target IOU for the untargeted attack increases success rates on all models even with 0.05 max-norm: The binned summaries and regression trendlines graph success proportion against target IOU for the untargeted attack in the randomized attack experiment. Bins are split into quantiles. Errors are 95% confidence intervals

```
bbox_iou_graph <- bbox_conf_data |> filter(target_or_perturb == "Target" & loss_target == "Untargeted")
bbox_iou_graph |>
  graph_attr(bbox_iou_eval, pred_name)
```

```
model <- partial(glm_model, predictor = "bbox_iou_eval")
data <- bbox_iou_graph

reg_est <- get_tidied_reg(model, data)
```

```
## `summarise()` has grouped output by 'model_name', 'loss_target'. You can
## override using the `.groups` argument.
```

```
ext_sig(reg_est, "neg")
```

```
## Total 5 predictors:
## 5 (100%) significant;
## 5 (100%) neg
```

```
## # A tibble: 5 x 9
## # Groups:   model_name, loss_target [5]
##   model_name    loss_target term   estimate std.error statistic p.value conf.low
##   <ord>         <ord>       <chr>     <dbl>     <dbl>     <dbl>   <dbl>    <dbl>
## 1 YOLOv3        Untargeted  bbox_~    -3.19     0.351     -9.10        0    -3.88
## 2 SSD           Untargeted  bbox_~    -2.75     0.288     -9.54        0    -3.31
## 3 RetinaNet     Untargeted  bbox_~    -3.08     0.328     -9.40        0    -3.72
## 4 Faster R-CNN  Untargeted  bbox_~    -2.02     0.374     -5.40        0    -2.74
## 5 Cascade R-CNN Untargeted  bbox_~    -2.90     0.364     -7.95        0    -3.61
## # i 1 more variable: conf.high <dbl>
```

```
print_statistics(reg_est, table_caption(pred_name, main_pt))
```

Table 8: We run a logistic model regressing success against target IOU for the untargeted attack in the randomized attack experiment. Target IOU for the untargeted attack increases success rates on all models. Table headers are explained in Appendix ??.

Group	Regression							
Model	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
Untargeted								
YOLOv3	bbox_iou_eval	*	-3.194	0.351	-9.098	0	-3.878	-2.501
SSD	bbox_iou_eval	*	-2.747	0.288	-9.539	0	-3.309	-2.180
RetinaNet	bbox_iou_eval	*	-3.085	0.328	-9.402	0	-3.725	-2.438
Faster R-CNN	bbox_iou_eval	*	-2.020	0.374	-5.403	0	-2.745	-1.278
Cascade R-CNN	bbox_iou_eval	*	-2.895	0.364	-7.953	0	-3.606	-2.177