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 ve
##
## 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                <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ 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> 11, 1, 7, 48, 53, 15, 10, 11, 46, 12, 14, 9, 1~
## $ vanish_count            <dbl> 3, 0, 2, 15, 34, 14, 9, 10, 39, 12, 14, 9, 17,~
## $ mislabel_count          <dbl> 8, 1, 5, 33, 19, 1, 1, 1, 7, 0, 0, 0, 0, 0, 0,~
## $ mislabel_intended_count <dbl> 8, 1, 4, 33, 19, 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("{bold_tex('Intent obfuscating attack is feasible for all models and attacks', params$norm)
```

```
## Warning in bold_tex("Intent obfuscating attack is feasible for all models and
## attacks", : NAs introduced by coercion
```

```
cap
```

```
## \textbf{Intent obfuscating attack is feasible for all models and attacks:}  We conduct a randomized
```

```
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                <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ 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> 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11~
## $ vanish_count            <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3~
## $ mislabel_count          <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8~
## $ mislabel_intended_count <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8~
## $ 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> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8~
## $ 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
```

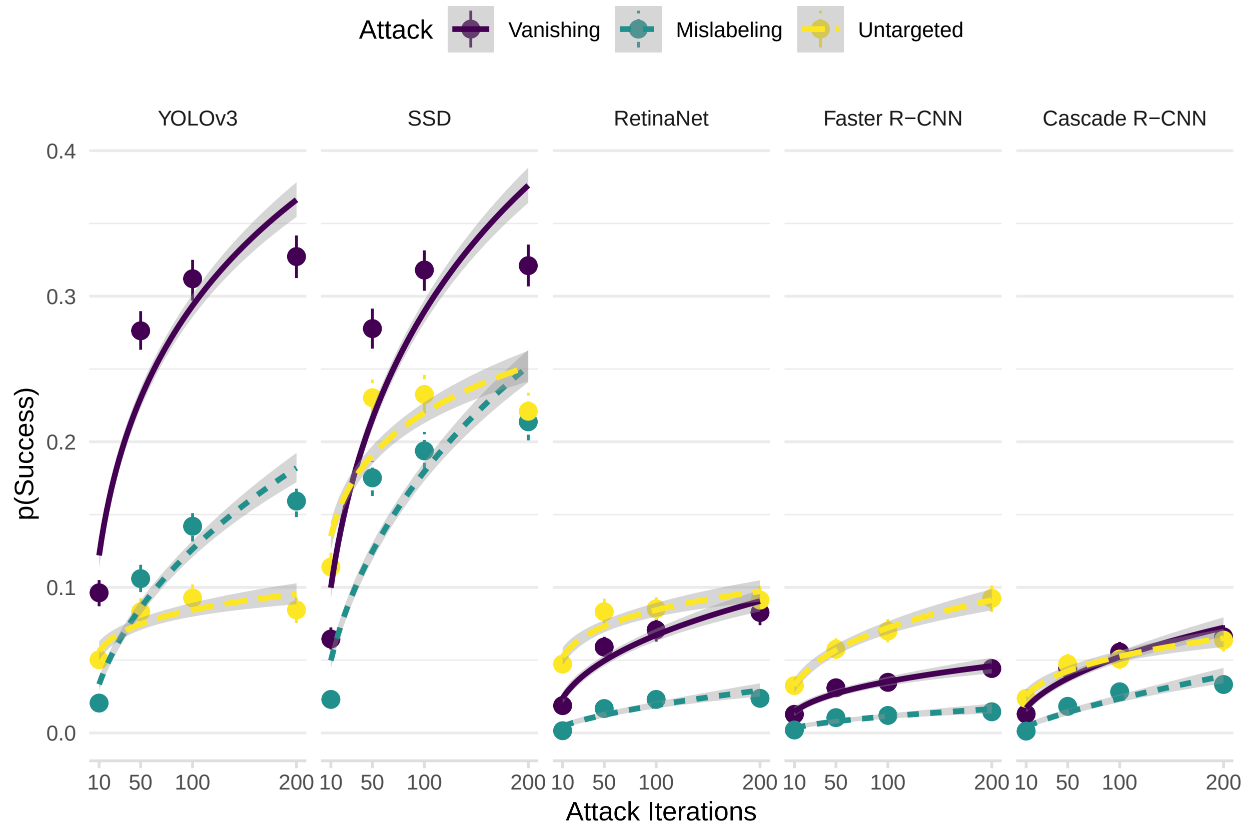


Figure 1: **Intent obfuscating attack is feasible for all models and attacks:** We conduct a randomized experiment by resampling COCO images, and within those images randomly sampling correctly predicted target and perturb objects. Then we distort the perturb objects to disrupt the target objects varying 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 **??**.

```
  # https://rdrr.io/cran/Hmisc/man/smean.sd.html
  stat_summary(fun.data = "mean_cl_boot") +
  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))
```

```
## Warning in norm_axy(params$norm): NAs introduced by coercion
```

```
# compare models against YOLO
# grouped by attack
data <- success_expanded_data |>
  # restrict to max iteration
  filter(num_iteration == max(num_iteration)) |>
```

```r
  # 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                <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ 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> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7~
## $ vanish_count            <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ mislabel_count          <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5~
## $ mislabel_intended_count <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5~
## $ 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> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5~
## $ success                 <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
```

```r
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.
```

```r
ext_sig(reg_est)
```

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

```
## # A tibble: 9 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_nam~    -1.68     0.067     -25.3       0    -1.82     -1.56
## 2 Vanishing   model_nam~   -2.35      0.084    -28.0        0   -2.52     -2.19
## 3 Vanishing   model_nam~   -1.93      0.072    -26.8        0   -2.07     -1.79
## 4 Mislabeling model_nam~    0.361     0.058      6.24       0    0.248     0.475
## 5 Mislabeling model_nam~   -2.05      0.112    -18.2        0   -2.28     -1.84
## 6 Mislabeling model_nam~   -2.56      0.139    -18.4        0   -2.84     -2.29
## 7 Mislabeling model_nam~   -1.71      0.098    -17.4        0   -1.90     -1.52
## 8 Untargeted  model_nam~    1.12      0.068     16.4        0    0.99      1.26
## 9 Untargeted  model_nam~   -0.304     0.086     -3.53       0   -0.474    -0.136
```

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

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		-0.029	0.048	-0.597	0.550	-0.122	0.065
	RetinaNet	*	-1.685	0.067	-25.317	0.000	-1.817	-1.556
	Faster R-CNN	*	-2.352	0.084	-28.021	0.000	-2.519	-2.190
	Cascade R-CNN	*	-1.929	0.072	-26.776	0.000	-2.072	-1.790
Mislabeling	YOLOv3		0.000					
	SSD	*	0.361	0.058	6.239	0.000	0.248	0.475
	RetinaNet	*	-2.052	0.112	-18.248	0.000	-2.278	-1.837
	Faster R-CNN	*	-2.555	0.139	-18.371	0.000	-2.838	-2.292
	Cascade R-CNN	*	-1.706	0.098	-17.372	0.000	-1.902	-1.517
Untargeted	YOLOv3		0.000					
	SSD	*	1.123	0.068	16.407	0.000	0.990	1.258
	RetinaNet		0.084	0.079	1.066	0.286	-0.071	0.239
	Faster R-CNN		0.099	0.079	1.259	0.208	-0.055	0.254
	Cascade R-CNN	*	-0.304	0.086	-3.531	0.000	-0.474	-0.136

```
# 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:
## 8 (53%) significant;
## 8 (53%) both
```

```
## # A tibble: 8 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~   -0.943     0.055    -17.2        0   -1.05     -0.836
## 2 YOLOv3        loss_ta~   -1.66      0.066    -25.2        0   -1.79     -1.53
## 3 SSD           loss_ta~   -0.553     0.051    -10.8        0   -0.654    -0.453
## 4 SSD           loss_ta~   -0.511     0.051    -10.0        0   -0.611    -0.411
## 5 RetinaNet     loss_ta~   -1.31      0.119    -11.0        0   -1.55     -1.08
## 6 Faster R-CNN  loss_ta~   -1.15      0.153     -7.49       0   -1.45     -0.853
## 7 Faster R-CNN  loss_ta~    0.789     0.094      8.37       0    0.606     0.976
## 8 Cascade R-CNN loss_ta~   -0.72      0.109     -6.62       0   -0.936    -0.509
```

```
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	*	-0.943	0.055	-17.212	0.000	-1.051	-0.836
	Untargeted	*	-1.662	0.066	-25.151	0.000	-1.793	-1.534
SSD	Vanishing		0.000					
	Mislabeling	*	-0.553	0.051	-10.779	0.000	-0.654	-0.453
	Untargeted	*	-0.511	0.051	-10.017	0.000	-0.611	-0.411
RetinaNet	Vanishing		0.000					
	Mislabeling	*	-1.311	0.119	-11.047	0.000	-1.548	-1.082
	Untargeted		0.107	0.079	1.348	0.178	-0.048	0.263
Faster R-CNN	Vanishing		0.000					
	Mislabeling	*	-1.146	0.153	-7.493	0.000	-1.454	-0.853
	Untargeted	*	0.789	0.094	8.370	0.000	0.606	0.976
	Vanishing		0.000					

Cascade R-CNN	Mislabeling	*	-0.720	0.109	-6.619	0.000	-0.936	-0.509
	Untargeted		-0.037	0.091	-0.409	0.683	-0.215	0.141

```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(~    0.476     0.019     25.3        0    0.439
##  2 YOLOv3        Mislabeling log(~    0.622     0.03      20.8        0    0.564
##  3 YOLOv3        Untargeted  log(~    0.192     0.028      6.78       0    0.137
##  4 SSD           Vanishing   log(~    0.566     0.02      28.5        0    0.527
##  5 SSD           Mislabeling log(~    0.621     0.025     24.5        0    0.572
##  6 SSD           Untargeted  log(~    0.256     0.019     13.4        0    0.219
##  7 RetinaNet     Vanishing   log(~    0.467     0.037     12.6        0    0.396
##  8 RetinaNet     Mislabeling log(~    0.635     0.076      8.33       0    0.49
##  9 RetinaNet     Untargeted  log(~    0.225     0.029      7.80       0    0.169
## 10 Faster R-CNN  Vanishing   log(~    0.397     0.049      8.16       0    0.303
## 11 Faster R-CNN  Mislabeling log(~    0.534     0.093      5.76       0    0.358
## 12 Faster R-CNN  Untargeted  log(~    0.367     0.034     10.9        0    0.302
## 13 Cascade R-CNN Vanishing   log(~    0.502     0.043     11.7        0    0.419
## 14 Cascade R-CNN Mislabeling log(~    0.753     0.073     10.3        0    0.613
## 15 Cascade R-CNN Untargeted  log(~    0.325     0.038      8.48       0    0.251
## # 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)	*	0.476	0.019	25.267	0	0.439	0.513
Mislabeling	log(iterations)	*	0.622	0.030	20.761	0	0.564	0.681
Untargeted	log(iterations)	*	0.192	0.028	6.776	0	0.137	0.247

SSD								
Vanishing	log(iterations)	*	0.566	0.020	28.456	0	0.527	0.605
Mislabeling	log(iterations)	*	0.621	0.025	24.466	0	0.572	0.672
Untargeted	log(iterations)	*	0.256	0.019	13.449	0	0.219	0.294
RetinaNet								
Vanishing	log(iterations)	*	0.467	0.037	12.620	0	0.396	0.541
Mislabeling	log(iterations)	*	0.635	0.076	8.331	0	0.490	0.789
Untargeted	log(iterations)	*	0.225	0.029	7.802	0	0.169	0.282
Faster R-CNN								
Vanishing	log(iterations)	*	0.397	0.049	8.160	0	0.303	0.494
Mislabeling	log(iterations)	*	0.534	0.093	5.762	0	0.358	0.722
Untargeted	log(iterations)	*	0.367	0.034	10.897	0	0.302	0.434
Cascade R-CNN								
Vanishing	log(iterations)	*	0.502	0.043	11.736	0	0.419	0.587
Mislabeling	log(iterations)	*	0.753	0.073	10.276	0	0.613	0.901
Untargeted	log(iterations)	*	0.325	0.038	8.477	0	0.251	0.401

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: 9,239,475
## 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> "dog", "dog", "dog", "dog", "dog", "dog", "~
## $ sample_mislabel_proba      <dbl> 2.556785e-05, 2.556785e-05, 2.556785e-05, 2~
## $ 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, 0.9556448, 0.901700~
## $ 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> FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FA~
## $ bbox_perturb               <lgl> FALSE, FALSE, FALSE, TRUE, FALSE, 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                   <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ 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> 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,~
## $ vanish_count               <dbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3~
## $ mislabel_count             <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8~
## $ mislabel_intended_count    <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8~
## $ 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,~
```

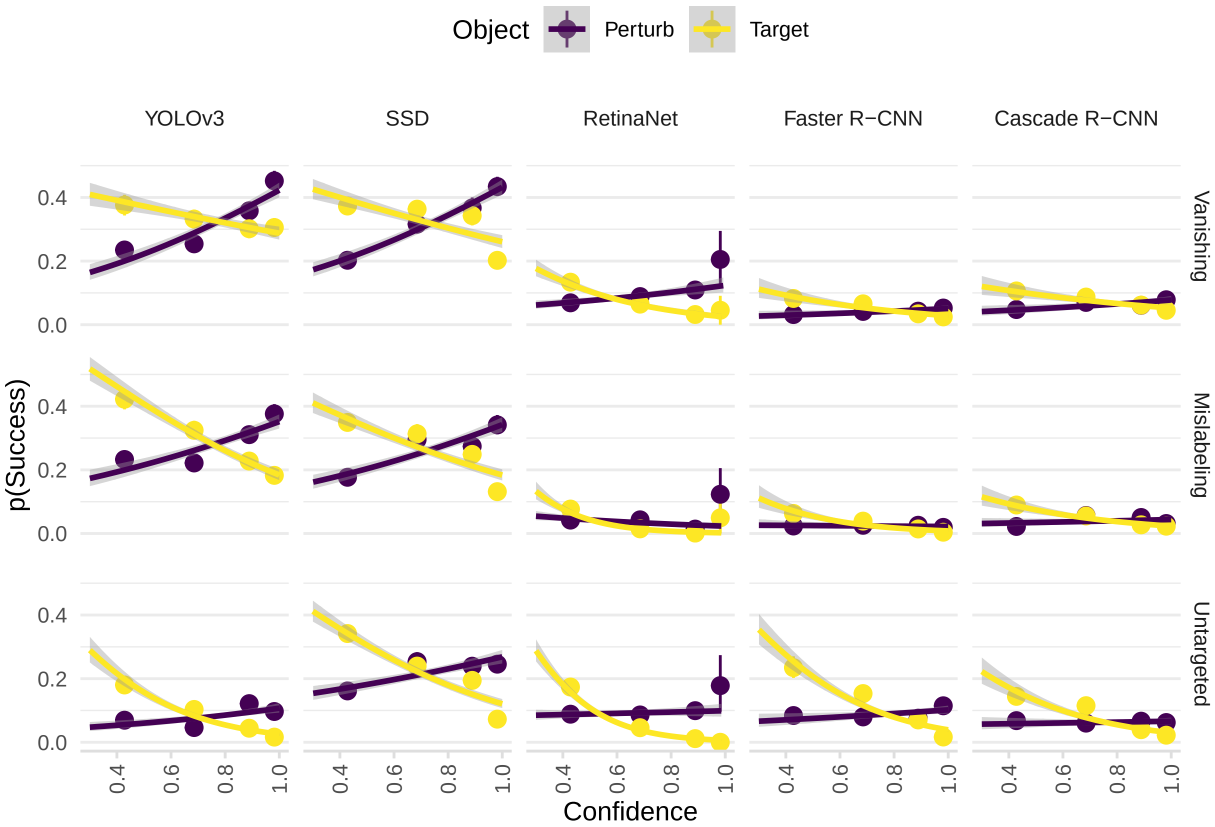
```r
# check whether target and perturb bboxes and
# mislabel classes are seeded across iterations
cols_start_equal(bbox_raw_data, c(
  "bbox_target", "bbox_perturb",
  "sample_mislabel_class", "sample_mislabel_proba"
))
```

```
## 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", "motorcycle", "surfboard", "cow", ~
## $ sample_mislabel_proba      <dbl> 6.615031e-05, 2.494136e-03, 4.392489e-05, 2~
## $ 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", "bicycle", "person", "elephant", "~
## $ bbox_xywhn                 <list<double>> <0.32723613, 0.26601949, 0.0435188~
## $ bbox_conf                  <dbl> 0.9305881, 0.6706054, 0.9882318, 0.9988155,~
## $ bbox_res_eval              <chr> "tp", "tp", "tp", "tp", "tp", "tp", "tp", "~
## $ bbox_iou_eval              <dbl> 0.8860679, 0.3753249, 0.9454082, 0.9255758,~
## $ bbox_res_pgd_eval          <chr> "tp", "tp", "fn", "tp", "tp", "tp", "tp", "~
## $ bbox_iou_pgd_eval          <dbl> 1.0000000, 1.0000000, NA, 0.8554562, 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                   <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ 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> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7~
## $ vanish_count               <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ mislabel_count             <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5~
## $ mislabel_intended_count    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5~
## $ 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~
```

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



```
# 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)
```

```
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.
```

```
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~   -0.773     0.153     -5.06       0    -1.07
##  2 YOLOv3        Mislabeling bbox~   -2.23      0.16     -13.9        0    -2.54
##  3 YOLOv3        Untargeted  bbox~   -3.91      0.268    -14.6        0    -4.44
```

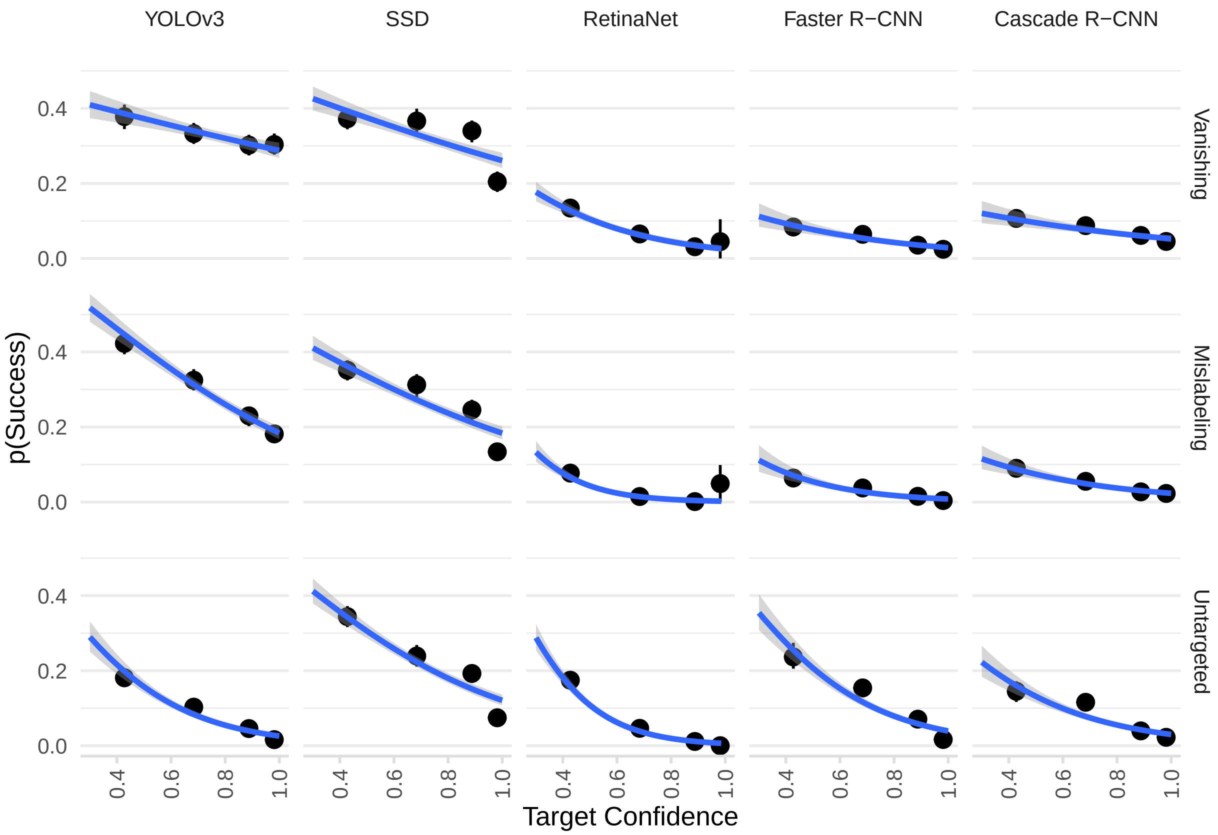


Figure 2: **Lower target confidence significantly increases success rates for all models and attacks:** The binned summaries and regression trendlines graph success proportion against target confidence in the randomized attack experiment. Bins are split into quantiles. Errors are 95% confidence intervals

```
##  4 SSD           Vanishing   bbox~   -1.06      0.142     -7.50        0    -1.34
##  5 SSD           Mislabeling bbox~   -1.62      0.151    -10.7         0    -1.91
##  6 SSD           Untargeted  bbox~   -2.33      0.164    -14.2         0    -2.65
##  7 RetinaNet     Vanishing   bbox~   -3.06      0.321     -9.54        0    -3.70
##  8 RetinaNet     Mislabeling bbox~   -6.13      0.616     -9.95        0    -7.39
##  9 RetinaNet     Untargeted  bbox~   -6.05      0.4      -15.1         0    -6.85
## 10 Faster R-CNN  Vanishing   bbox~   -2.08      0.326     -6.38        0    -2.71
## 11 Faster R-CNN  Mislabeling bbox~   -3.90      0.449     -8.70        0    -4.80
## 12 Faster R-CNN  Untargeted  bbox~   -3.72      0.239    -15.6         0    -4.19
## 13 Cascade R-CNN Vanishing   bbox~   -1.30      0.275     -4.73        0    -1.83
## 14 Cascade R-CNN Mislabeling bbox~   -2.43      0.332     -7.32        0    -3.08
## 15 Cascade R-CNN Untargeted  bbox~   -3.18      0.271    -11.7         0    -3.72
## # 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	*	-0.773	0.153	-5.059	0	-1.072	-0.473
Mislabeling	confidence	*	-2.230	0.160	-13.915	0	-2.545	-1.917
Untargeted	confidence	*	-3.910	0.268	-14.579	0	-4.442	-3.390
SSD								
Vanishing	confidence	*	-1.063	0.142	-7.505	0	-1.341	-0.786
Mislabeling	confidence	*	-1.616	0.151	-10.714	0	-1.913	-1.321
Untargeted	confidence	*	-2.326	0.164	-14.203	0	-2.649	-2.007
RetinaNet								
Vanishing	confidence	*	-3.057	0.321	-9.535	0	-3.695	-2.437
Mislabeling	confidence	*	-6.133	0.616	-9.952	0	-7.389	-4.969
Untargeted	confidence	*	-6.050	0.400	-15.130	0	-6.853	-5.284
Faster R-CNN								
Vanishing	confidence	*	-2.079	0.326	-6.383	0	-2.714	-1.436
Mislabeling	confidence	*	-3.903	0.449	-8.702	0	-4.795	-3.032
Untargeted	confidence	*	-3.719	0.239	-15.564	0	-4.190	-3.253
Cascade R-CNN								
Vanishing	confidence	*	-1.298	0.275	-4.727	0	-1.831	-0.754
Mislabeling	confidence	*	-2.428	0.332	-7.317	0	-3.077	-1.775
Untargeted	confidence	*	-3.183	0.271	-11.740	0	-3.716	-2.653

```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()
```

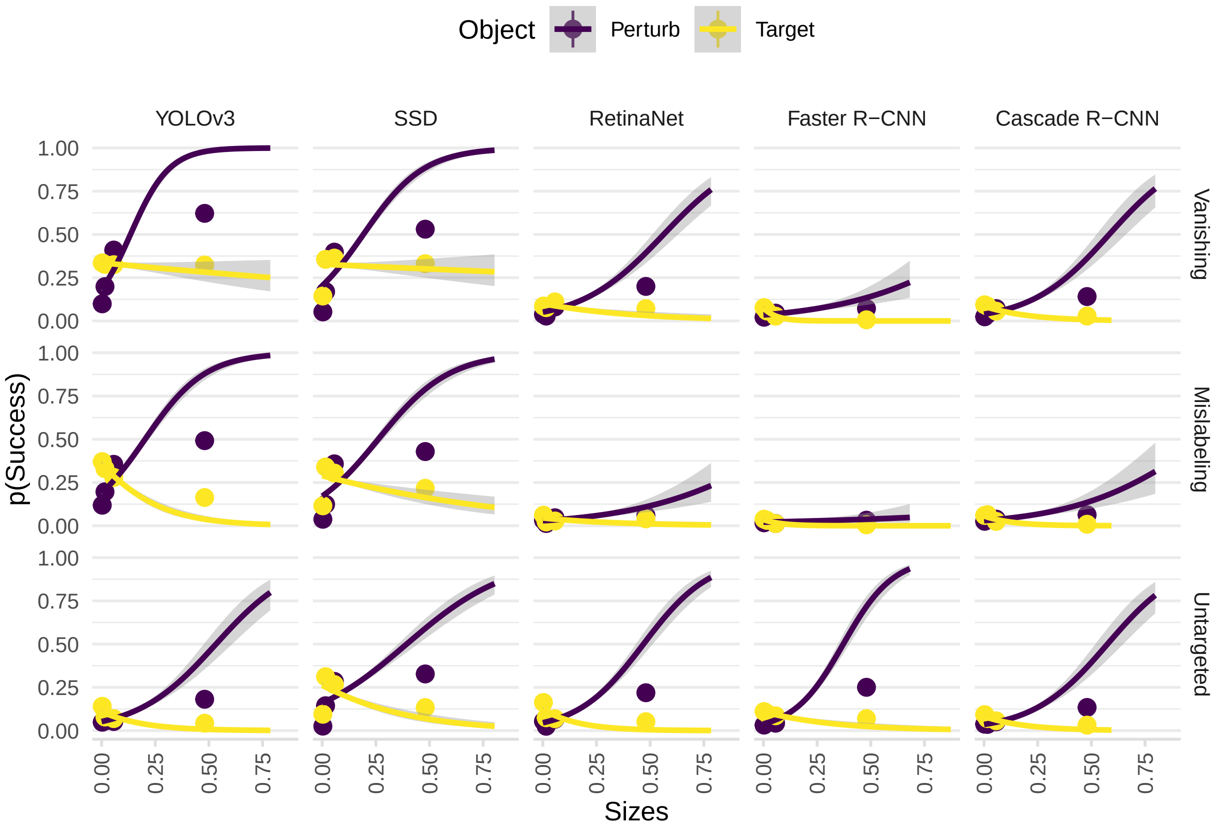
```
## `summarise()` has grouped output by 'model_name'. You can override using the
## `.groups` argument.
```

```
## 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> 0.67200, 0.71275, 0.91550, 0.67675, 0.73375, 0.77900, 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", "motorcycle", "surfboard", "cow", ~
## $ sample_mislabel_proba      <dbl> 6.615031e-05, 2.494136e-03, 4.392489e-05, 2~
## $ 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", "bicycle", "person", "elephant", "~
## $ bbox_xn                    <dbl> 0.32723613, 0.15173593, 0.37364487, 0.31803~
## $ bbox_yn                    <dbl> 0.26601949, 0.52290444, 0.31231453, 0.16039~
## $ bbox_wn                    <dbl> 0.04351888, 0.07043431, 0.35480569, 0.33195~
## $ bbox_hn                    <dbl> 0.10756386, 0.04831026, 0.67813552, 0.80579~
## $ bbox_conf                  <dbl> 0.9305881, 0.6706054, 0.9882318, 0.9988155,~
## $ bbox_res_eval              <chr> "tp", "tp", "tp", "tp", "tp", "tp", "tp", "~
## $ bbox_iou_eval              <dbl> 0.8860679, 0.3753249, 0.9454082, 0.9255758,~
## $ bbox_res_pgd_eval          <chr> "tp", "tp", "fn", "tp", "tp", "tp", "tp", "~
## $ bbox_iou_pgd_eval          <dbl> 1.0000000, 1.0000000, NA, 0.8554562, 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                   <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ 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> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7~
## $ vanish_count               <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ mislabel_count             <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5~
## $ mislabel_intended_count    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5~
## $ 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, 0.0034026994, 0.2406063427, 0~
```

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

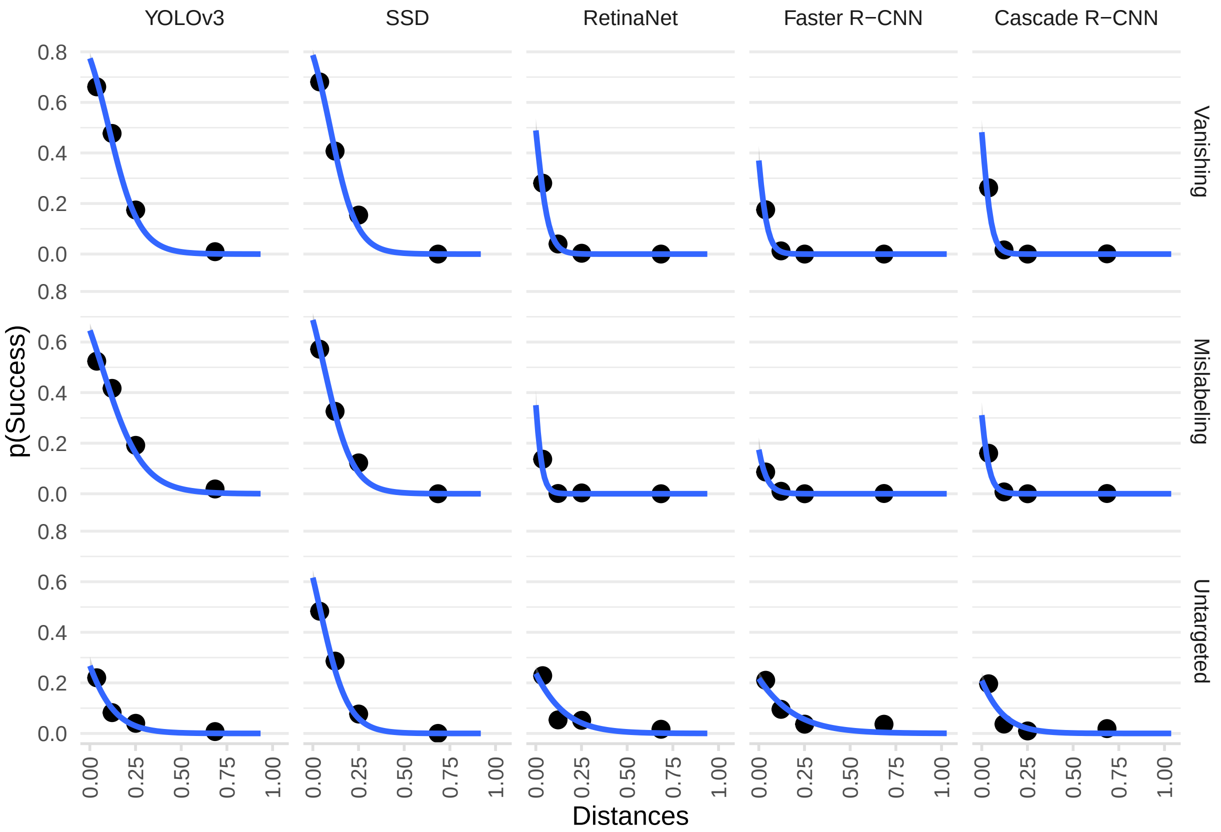


```
# bbox distances typically based on ground-truth attacked bbox as in sizes
bbox_dist_data <- bbox_size_data |>
  mutate(
    target_or_perturb_lower = str_to_lower(target_or_perturb)
  ) |>
  # mainly "group" by sample_id and attack iteration
  # with target bbox on one row and perturb on another
  # 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, 0.15173593, 0.37364487, 0.31803055, 0.89~
## $ bbox_xn_perturb   <dbl> 6.392759e-01, 6.121438e-01, 3.132355e-02, 1.640413e-~
## $ bbox_yn_target    <dbl> 0.26601949, 0.52290444, 0.31231453, 0.16039687, 0.19~
## $ bbox_yn_perturb   <dbl> 0.78020687, 0.50323486, 0.77699087, 0.45312066, 0.50~
## $ bbox_wn_target    <dbl> 0.04351888, 0.07043431, 0.35480569, 0.33195910, 0.06~
## $ bbox_wn_perturb   <dbl> 0.02060528, 0.05746603, 0.18098172, 0.13549221, 0.13~
## $ bbox_hn_target    <dbl> 0.10756386, 0.04831026, 0.67813552, 0.80579662, 0.03~
## $ bbox_hn_perturb   <dbl> 0.08544267, 0.03637395, 0.21702971, 0.54033688, 0.05~
## $ bbox_size_target  <dbl> 0.0046810584, 0.0034026994, 0.2406063427, 0.26749152~
## $ bbox_size_perturb <dbl> 0.0017605700, 0.0020902666, 0.0392784101, 0.07321143~
## $ bbox_dist         <dbl> 0.48728447, 0.38997352, 0.16133960, 0.01849709, 0.46~
```

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



```
saveRDS(bbox_dist_data, here("analysis/rand_dist_size.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}")
```

```r
main_pt <- "Larger perturb objects significantly increase success rates for all models and attacks, exc

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

```
## Warning in bold_tex(main_pt, params$norm): NAs introduced by coercion
```

```r
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 5 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 4 remaining warnings.
```

```
## Warning: There were 212 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 211 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.
```

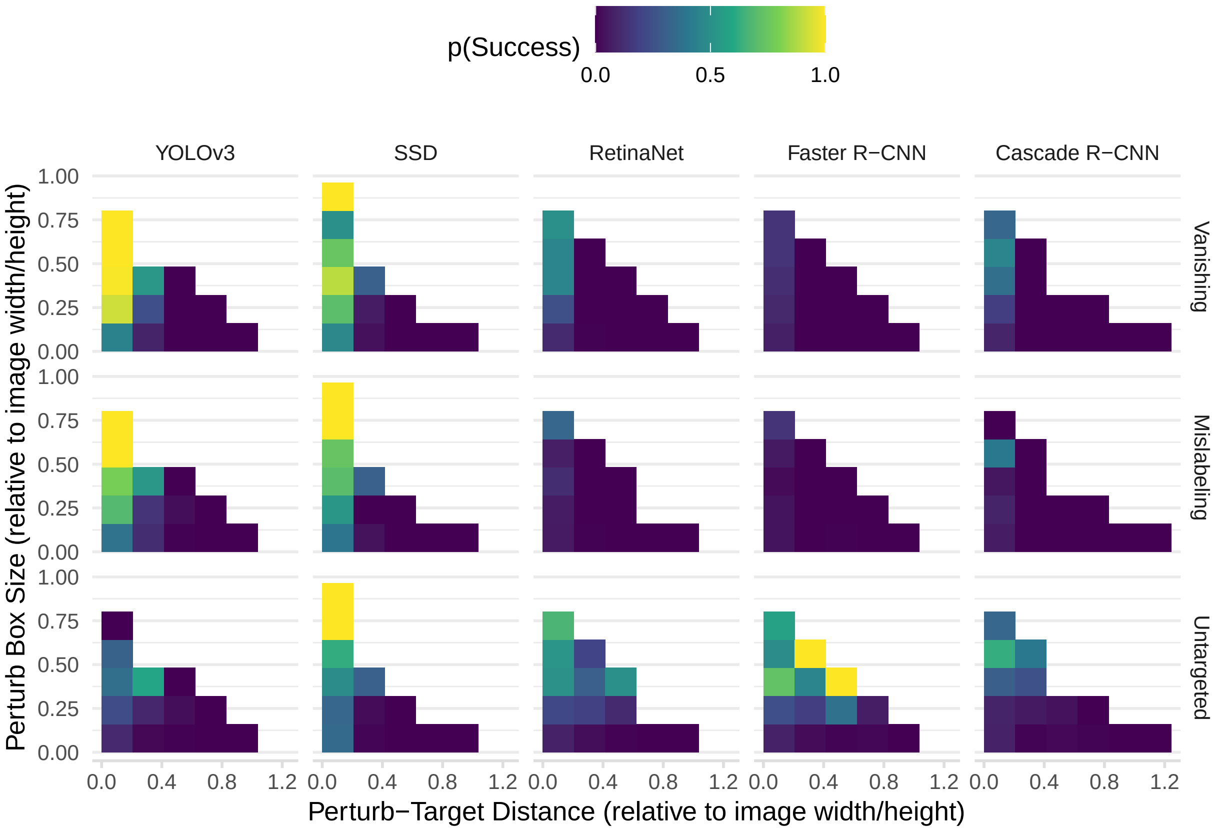


Figure 3: **Larger perturb objects significantly increase success rates for all models and attacks, except for mislabeling attack on Faster R-CNN, after controlling for perturb-target distances. Shorter perturb-target distances significantly increase success rates for all models and attacks, after controlling for perturb object sizes:** 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.

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

```
## 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_perturb
##   .. ..- 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_perturb
##   .. ..- 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] 1312 1312 1312 1149 1149 ...
##   .. ..- attr(*, "names")= chr [1:45] "bbox_dist" "bbox_size_perturb" "bbox_dist:bbox_size_perturb"
##   ..$ estimate      : num [1:45] -9.67 32.88 -96.58 -8.32 8.23 ...
##   ..$ std.error     : num [1:45] 0.656 2.2 10.405 0.516 0.837 ...
##   ..$ statistic     : num [1:45] -14.74 14.94 -9.28 -16.12 9.83 ...
##   ..$ p.value       : num [1:45] 3.65e-49 1.68e-50 1.66e-20 1.80e-58 8.13e-23 ...
##   ..$ conf.low      : num [1:45] -10.99 28.7 -117.51 -9.35 6.64 ...
##   ..$ conf.high     : num [1:45] -8.41 37.32 -76.73 -7.33 9.92 ...
##   ..- 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~    -9.67     0.656    -14.7        0   -11.0
##  2 YOLOv3        Mislabeling bbox~    -8.32     0.516    -16.1        0    -9.36
##  3 YOLOv3        Untargeted  bbox~   -13.3      1.15     -11.6        0   -15.6
##  4 SSD           Vanishing   bbox~   -14.4      0.758    -19.0        0   -15.9
##  5 SSD           Mislabeling bbox~   -12.0      0.729    -16.5        0   -13.5
##  6 SSD           Untargeted  bbox~   -14.1      0.811    -17.4        0   -15.8
##  7 RetinaNet     Vanishing   bbox~   -38.7      2.84     -13.6        0   -44.4
##  8 RetinaNet     Mislabeling bbox~   -48.1      5.19      -9.28       0   -58.8
##  9 RetinaNet     Untargeted  bbox~   -13.2      1.19     -11.1        0   -15.6
## 10 Faster R-CNN  Vanishing   bbox~   -31.5      3.27      -9.62       0   -38.2
## 11 Faster R-CNN  Mislabeling bbox~   -24.3      3.51      -6.91       0   -31.6
## 12 Faster R-CNN  Untargeted  bbox~   -14.4      1.24     -11.6        0   -16.9
## 13 Cascade R-CNN Vanishing   bbox~   -27.7      2.84      -9.78       0   -33.6
## 14 Cascade R-CNN Mislabeling bbox~   -28.7      3.36      -8.53       0   -35.7
## 15 Cascade R-CNN Untargeted  bbox~   -13.4      1.30     -10.3        0   -16.1
## # i 1 more variable: conf.high <dbl>
```

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

```
## ----------bbox_size_perturb----------
## Total 15 predictors:
## 14 (93%) significant;
## 14 (93%) pos
## # A tibble: 14 x 9
## # Groups:   model_name, loss_target [14]
##    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~    32.9      2.2       14.9     0        28.7
##  2 YOLOv3        Mislabeling bbox~     8.23     0.837      9.83    0         6.64
##  3 YOLOv3        Untargeted  bbox~     1.64     0.647      2.53    0.011     0.369
##  4 SSD           Vanishing   bbox~     9.33     0.959      9.73    0         7.51
##  5 SSD           Mislabeling bbox~     7.73     0.806      9.59    0         6.20
##  6 SSD           Untargeted  bbox~     2.30     0.528      4.35    0         1.29
##  7 RetinaNet     Vanishing   bbox~     1.92     0.675      2.84    0.005     0.647
##  8 RetinaNet     Mislabeling bbox~     2.27     1.15       1.97    0.049     0.074
##  9 RetinaNet     Untargeted  bbox~     2.54     0.519      4.89    0         1.53
## 10 Faster R-CNN  Vanishing   bbox~     3.76     1.09       3.46    0.001     1.68
## 11 Faster R-CNN  Untargeted  bbox~     2.18     0.65       3.36    0.001     0.913
## 12 Cascade R-CNN Vanishing   bbox~     7.19     0.906      7.94    0         5.49
## 13 Cascade R-CNN Mislabeling bbox~     2.58     0.763      3.39    0.001     1.09
## 14 Cascade R-CNN Untargeted  bbox~     2.59     0.561      4.62    0         1.49
## # 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:
## 11 (73%) significant;
## 11 (73%) both
```

```
## # 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~   -96.6      10.4      -9.28    0      -118.
##  2 YOLOv3        Mislabeling bbox~    -9.86      4.88     -2.02    0.043   -19.7
##  3 YOLOv3        Untargeted  bbox~    31.6       5.86      5.39    0        20.0
##  4 SSD           Mislabeling bbox~   -13.6       5.56     -2.45    0.014   -24.8
##  5 SSD           Untargeted  bbox~    11.9       4.57      2.61    0.009     2.78
##  6 RetinaNet     Vanishing   bbox~    53.2      10.7       4.95    0        31.2
##  7 RetinaNet     Untargeted  bbox~    36.0       4.72      7.63    0        27.0
##  8 Faster R-CNN  Untargeted  bbox~    58.7       5.96      9.85    0        47.3
##  9 Cascade R-CNN Vanishing   bbox~   -77.4      22.6      -3.43    0.001  -125.
## 10 Cascade R-CNN Mislabeling bbox~   -69.6      31.2      -2.23    0.026  -136.
## 11 Cascade R-CNN Untargeted  bbox~    25.3       4.98      5.08    0        15.5
## # 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	*	-9.672	0.656	-14.738	0.000	-10.986	-8.413
	size	*	32.877	2.200	14.945	0.000	28.697	37.320
	distance * size	*	-96.578	10.405	-9.282	0.000	-117.509	-76.730
Mislabeling	distance	*	-8.322	0.516	-16.121	0.000	-9.355	-7.331
	size	*	8.229	0.837	9.833	0.000	6.635	9.917
	distance * size	*	-9.864	4.876	-2.023	0.043	-19.658	-0.531
Untargeted	distance	*	-13.317	1.151	-11.566	0.000	-15.649	-11.136
	size	*	1.638	0.647	2.532	0.011	0.369	2.909
	distance * size	*	31.584	5.862	5.388	0.000	20.028	43.048
SSD								
Vanishing	distance	*	-14.374	0.758	-18.971	0.000	-15.892	-12.921
	size	*	9.330	0.959	9.729	0.000	7.508	11.267
	distance * size		-7.647	5.626	-1.359	0.174	-18.998	3.079
Mislabeling	distance	*	-12.008	0.729	-16.468	0.000	-13.473	-10.614
	size	*	7.727	0.806	9.591	0.000	6.198	9.357
	distance * size	*	-13.614	5.556	-2.451	0.014	-24.820	-3.030
Untargeted	distance	*	-14.125	0.811	-17.425	0.000	-15.757	-12.579
	size	*	2.298	0.528	4.353	0.000	1.289	3.361
	distance * size	*	11.937	4.573	2.611	0.009	2.779	20.724
RetinaNet								
Vanishing	distance	*	-38.670	2.842	-13.608	0.000	-44.429	-33.288
	size	*	1.917	0.675	2.840	0.005	0.647	3.291
	distance * size	*	53.194	10.742	4.952	0.000	31.190	73.157
Mislabeling	distance	*	-48.140	5.186	-9.283	0.000	-58.781	-38.448
	size	*	2.270	1.151	1.972	0.049	0.074	4.594
	distance * size		7.234	25.556	0.283	0.777	-46.376	53.609
Untargeted	distance	*	-13.171	1.189	-11.082	0.000	-15.598	-10.938
	size	*	2.541	0.519	4.892	0.000	1.526	3.565
	distance * size	*	36.039	4.724	7.629	0.000	27.007	45.549
Faster R-CNN								
Vanishing	distance	*	-31.462	3.270	-9.622	0.000	-38.181	-25.358
	size	*	3.758	1.086	3.462	0.001	1.675	5.942
	distance * size		-35.320	23.347	-1.513	0.130	-84.636	7.187
Mislabeling	distance	*	-24.289	3.513	-6.914	0.000	-31.624	-17.853
	size		1.648	1.414	1.166	0.244	-1.207	4.385
	distance * size		-37.467	32.660	-1.147	0.251	-108.916	19.888
Untargeted	distance	*	-14.429	1.244	-11.603	0.000	-16.949	-12.074
	size	*	2.184	0.650	3.360	0.001	0.913	3.465
	distance * size	*	58.694	5.959	9.849	0.000	47.273	70.648

Cascade R-CNN								
Vanishing	distance	*	-27.740	2.837	-9.778	0.000	-33.578	-22.453
	size	*	7.189	0.906	7.936	0.000	5.488	9.045
	distance * size	*	-77.368	22.567	-3.428	0.001	-125.142	-36.519
Mislabeling	distance	*	-28.681	3.361	-8.533	0.000	-35.680	-22.493
	size	*	2.584	0.763	3.388	0.001	1.094	4.093
	distance * size	*	-69.647	31.193	-2.233	0.026	-136.025	-13.985
Untargeted	distance	*	-13.415	1.297	-10.340	0.000	-16.058	-10.972
	size	*	2.594	0.561	4.621	0.000	1.492	3.697
	distance * size	*	25.276	4.976	5.079	0.000	15.453	35.061

```
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> 0.0000411684, 0.0081496051, 0.0162580419, 0.02436647~
```

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

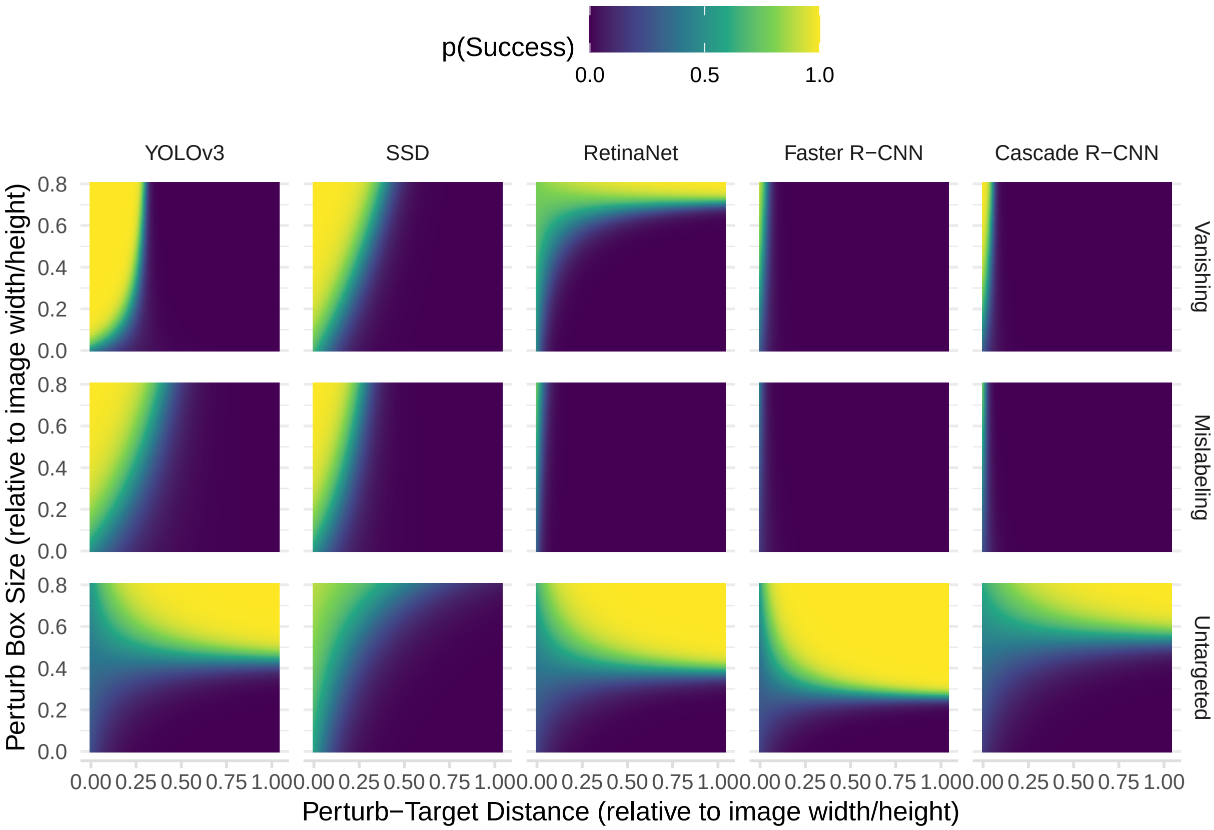
```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `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> 0.0000411684, 0.0081496051, 0.0162580419, 0.02436647~
## $ success           <dbl> 0.4717830, 0.5383202, 0.6035197, 0.6652374, 0.721776~
```

```
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()
```

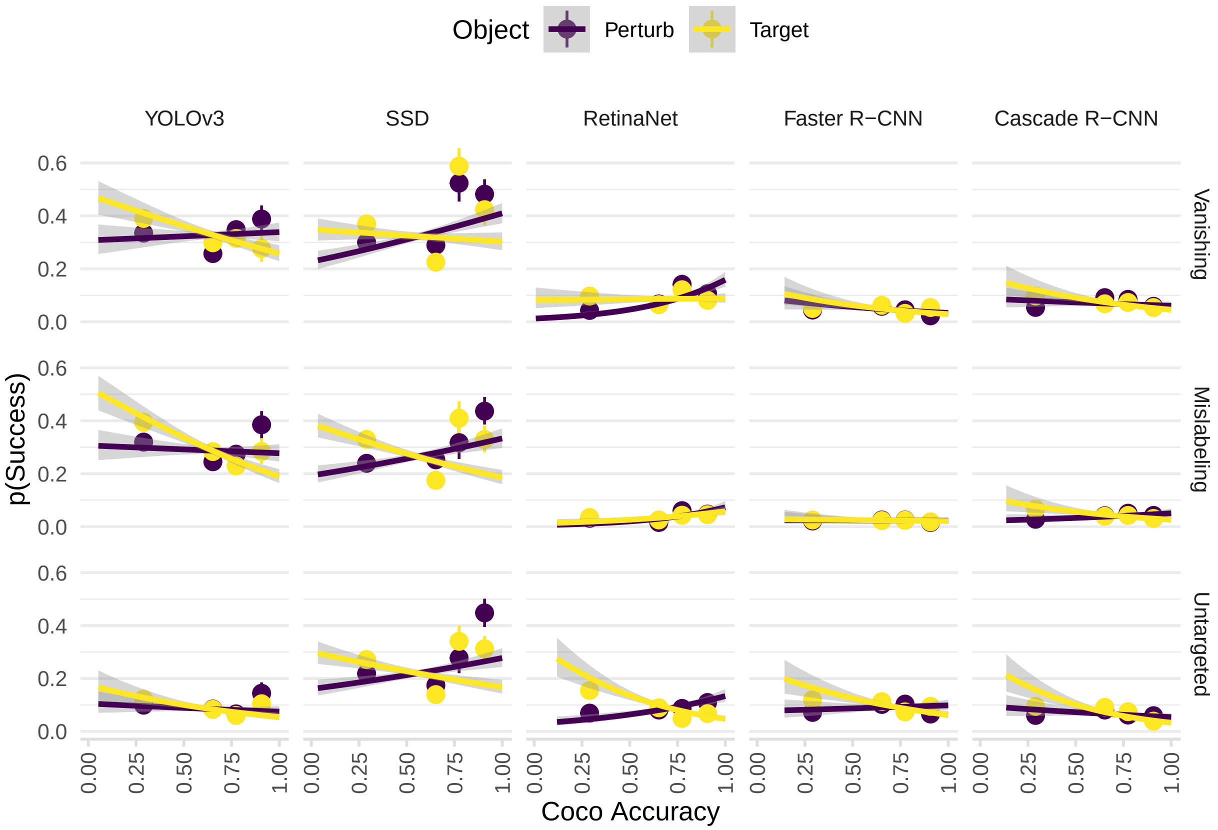
```
## Rows: 382
## Columns: 3
## $ model_name   <ord> YOLOv3, YOLOv3, YOLOv3, YOLOv3, YOLOv3, YOLOv3, YOLOv3, Y~
## $ bbox_class   <chr> "airplane", "apple", "backpack", "banana", "baseball bat"~
## $ gt_p_success <dbl> 1.0000000, 0.7222222, 0.2941176, 0.1111111, 0.5000000, 0.~
```

```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", "motorcycle", "surfboard", "cow", ~
## $ sample_mislabel_proba      <dbl> 6.615031e-05, 2.494136e-03, 4.392489e-05, 2~
## $ 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", "bicycle", "person", "elephant", "~
## $ bbox_xywhn                 <list<double>> <0.32723613, 0.26601949, 0.0435188~
## $ bbox_conf                  <dbl> 0.9305881, 0.6706054, 0.9882318, 0.9988155,~
## $ bbox_res_eval              <chr> "tp", "tp", "tp", "tp", "tp", "tp", "tp", "~
## $ bbox_iou_eval              <dbl> 0.8860679, 0.3753249, 0.9454082, 0.9255758,~
## $ bbox_res_pgd_eval          <chr> "tp", "tp", "fn", "tp", "tp", "tp", "tp", "~
## $ bbox_iou_pgd_eval          <dbl> 1.0000000, 1.0000000, NA, 0.8554562, 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                   <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ 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> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7~
## $ vanish_count               <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2~
## $ mislabel_count             <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5~
## $ mislabel_intended_count    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5~
## $ 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, 0.6000000, 0.8199719, 1.0000000,~
```

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



```r
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_
```

```
## Warning in bold_tex(str_to_sentence(main_pt), norm): NAs introduced by coercion
```

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

```r
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.
```

```r
# 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")
```

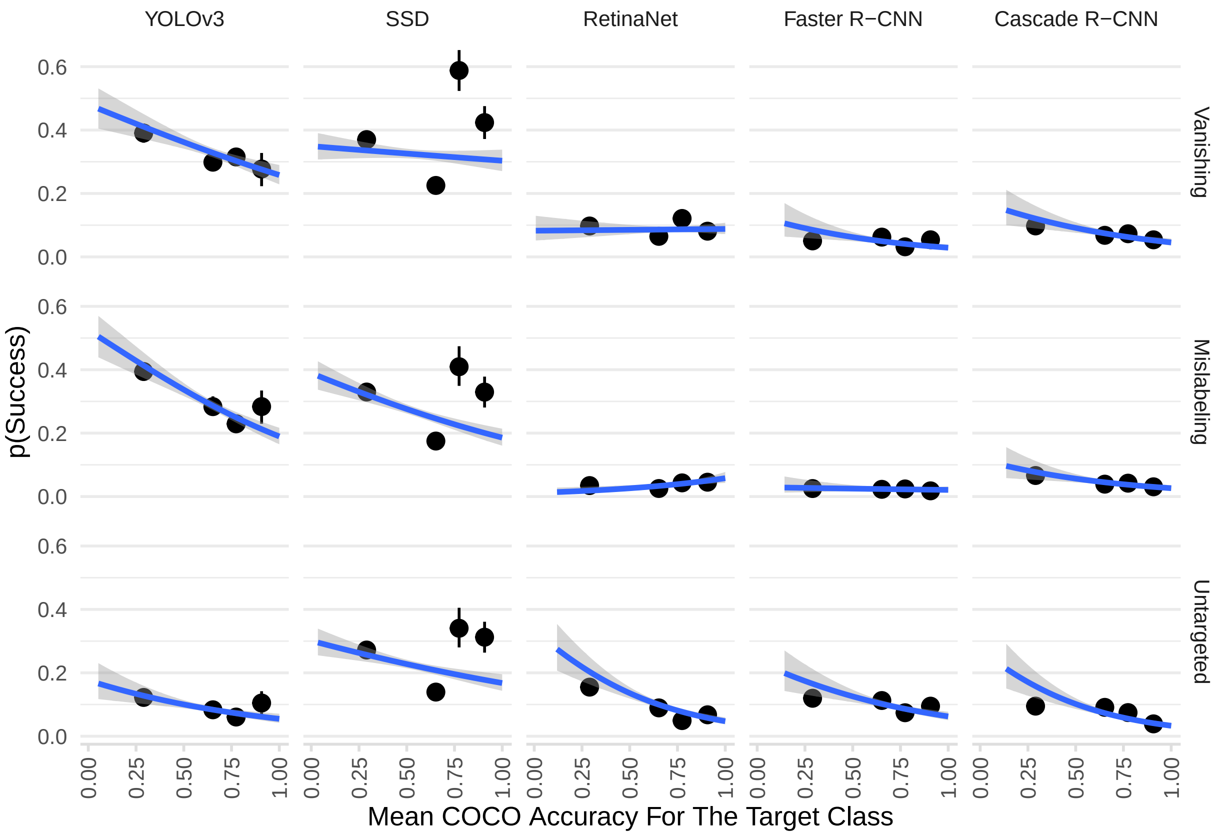


Figure 4: **Although higher mean COCO accuracy for the target class seem to decrease success rates, the results are mixed after controlling for target class confidence (Table 6):** The binned summaries and regression trendlines graph success proportion against mean COCO accuracy for the target class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

```
## ----------gt_p_success----------
## Total 15 predictors:
## 7 (47%) significant;
## 3 (20%) neg

## # A tibble: 3 x 9
## # Groups:   model_name, loss_target [3]
##   model_name     loss_target term    estimate std.error statistic p.value conf.low
##   <ord>          <ord>       <chr>      <dbl>     <dbl>     <dbl>   <dbl>    <dbl>
## 1 Faster R-CNN   Vanishing   gt_p_~     -5.57      1.54     -3.61   0        -8.59
## 2 Faster R-CNN   Untargeted  gt_p_~     -3.04      1.15     -2.65   0.008    -5.30
## 3 Cascade R-CNN Vanishing   gt_p_~     -3.47      1.41     -2.47   0.014    -6.22
## # i 1 more variable: conf.high <dbl>
```

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

```
## ----------gt_p_success----------
## Total 15 predictors:
## 7 (47%) significant;
## 4 (27%) pos

## # A tibble: 4 x 9
## # Groups:   model_name, loss_target [4]
##   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~     1.28     0.511      2.51   0.012    0.283
## 2 SSD        Mislabeling gt_p_suc~     3.28     0.549      5.98   0        2.21
## 3 SSD        Untargeted  gt_p_suc~     4.52     0.584      7.74   0        3.38
## 4 RetinaNet  Untargeted  gt_p_suc~     2.47     1.21       2.05   0.04     0.109
## # i 1 more variable: conf.high <dbl>
```

```
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_~    -2.20     0.976     -2.25   0.024   -4.11
##  2 YOLOv3       Mislabeling gt_p_~    -3.37     1.02      -3.29   0.001   -5.38
##  3 YOLOv3       Untargeted  gt_p_~    -3.38     1.70      -1.99   0.047   -6.70
##  4 SSD          Vanishing   gt_p_~    -1.91     0.71      -2.68   0.007   -3.30
##  5 SSD          Mislabeling gt_p_~    -6.18     0.795     -7.77   0       -7.75
##  6 SSD          Untargeted  gt_p_~    -7.78     0.874     -8.90   0       -9.51
##  7 RetinaNet    Untargeted  gt_p_~    -6.67     2.55      -2.61   0.009  -11.7
##  8 Faster R-CNN Vanishing   gt_p_~     6.50     2.13       3.05   0.002    2.33
##  9 Faster R-CNN Mislabeling gt_p_~     8.37     3.36       2.49   0.013    1.78
## 10 Faster R-CNN Untargeted  gt_p_~     3.93     1.67       2.35   0.019    0.676
## # i 1 more variable: conf.high <dbl>
```

```
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		0.726	0.732	0.992	0.321	-0.707	2.164
	confidence		0.733	0.652	1.124	0.261	-0.544	2.014
	accuracy * confidence	*	-2.196	0.976	-2.250	0.024	-4.113	-0.285
Mislabeling	accuracy		1.133	0.743	1.524	0.128	-0.325	2.591
	confidence		0.044	0.679	0.065	0.948	-1.289	1.373
	accuracy * confidence	*	-3.371	1.025	-3.289	0.001	-5.382	-1.363
Untargeted	accuracy		1.324	1.060	1.248	0.212	-0.749	3.410

		confidence		-1.696	1.113	-1.525	0.127	-3.895	0.469
		accuracy * confidence	*	-3.376	1.697	-1.989	0.047	-6.701	-0.047
SSD									
	Vanishing	accuracy	*	1.282	0.511	2.508	0.012	0.283	2.288
		confidence		0.017	0.426	0.040	0.968	-0.816	0.854
		accuracy * confidence	*	-1.907	0.710	-2.684	0.007	-3.304	-0.519
	Mislabeling	accuracy	*	3.281	0.549	5.976	0.000	2.210	4.363
		confidence	*	1.871	0.460	4.067	0.000	0.972	2.776
		accuracy * confidence	*	-6.178	0.795	-7.769	0.000	-7.747	-4.629
	Untargeted	accuracy	*	4.517	0.584	7.738	0.000	3.381	5.670
		confidence	*	1.990	0.499	3.985	0.000	1.014	2.971
		accuracy * confidence	*	-7.783	0.874	-8.905	0.000	-9.508	-6.081
RetinaNet									
	Vanishing	accuracy		1.009	1.143	0.883	0.377	-1.217	3.262
		confidence	*	-3.823	1.744	-2.192	0.028	-7.277	-0.442
		accuracy * confidence		0.571	2.246	0.254	0.799	-3.819	4.984
	Mislabeling	accuracy		2.565	2.044	1.255	0.209	-1.385	6.612
		confidence	*	-8.994	3.794	-2.371	0.018	-16.549	-1.716
		accuracy * confidence		2.506	4.691	0.534	0.593	-6.650	11.687
	Untargeted	accuracy	*	2.471	1.206	2.049	0.040	0.109	4.837
		confidence		-1.214	1.810	-0.671	0.503	-4.820	2.279
		accuracy * confidence	*	-6.672	2.553	-2.613	0.009	-11.666	-1.654
Faster R-CNN									
	Vanishing	accuracy	*	-5.572	1.544	-3.608	0.000	-8.586	-2.520
		confidence	*	-6.548	1.557	-4.206	0.000	-9.623	-3.513
		accuracy * confidence	*	6.505	2.134	3.047	0.002	2.327	10.700
	Mislabeling	accuracy		-4.008	2.072	-1.935	0.053	-7.990	0.140
		confidence	*	-10.366	2.631	-3.940	0.000	-15.562	-5.263
		accuracy * confidence	*	8.374	3.358	2.494	0.013	1.781	14.920
	Untargeted	accuracy	*	-3.045	1.151	-2.646	0.008	-5.305	-0.788
		confidence	*	-6.522	1.247	-5.229	0.000	-8.997	-4.105
		accuracy * confidence	*	3.928	1.670	2.353	0.019	0.676	7.222
Cascade R-CNN									
	Vanishing	accuracy	*	-3.474	1.409	-2.466	0.014	-6.223	-0.691
		confidence	*	-3.241	1.281	-2.530	0.011	-5.742	-0.712
		accuracy * confidence		3.012	1.787	1.685	0.092	-0.505	6.509
	Mislabeling	accuracy		-2.849	1.600	-1.780	0.075	-5.961	0.326
		confidence	*	-4.204	1.580	-2.661	0.008	-7.303	-1.099
		accuracy * confidence		2.670	2.171	1.229	0.219	-1.600	6.920
	Untargeted	accuracy		-0.996	1.283	-0.776	0.438	-3.504	1.532
		confidence		-2.287	1.256	-1.821	0.069	-4.759	0.171

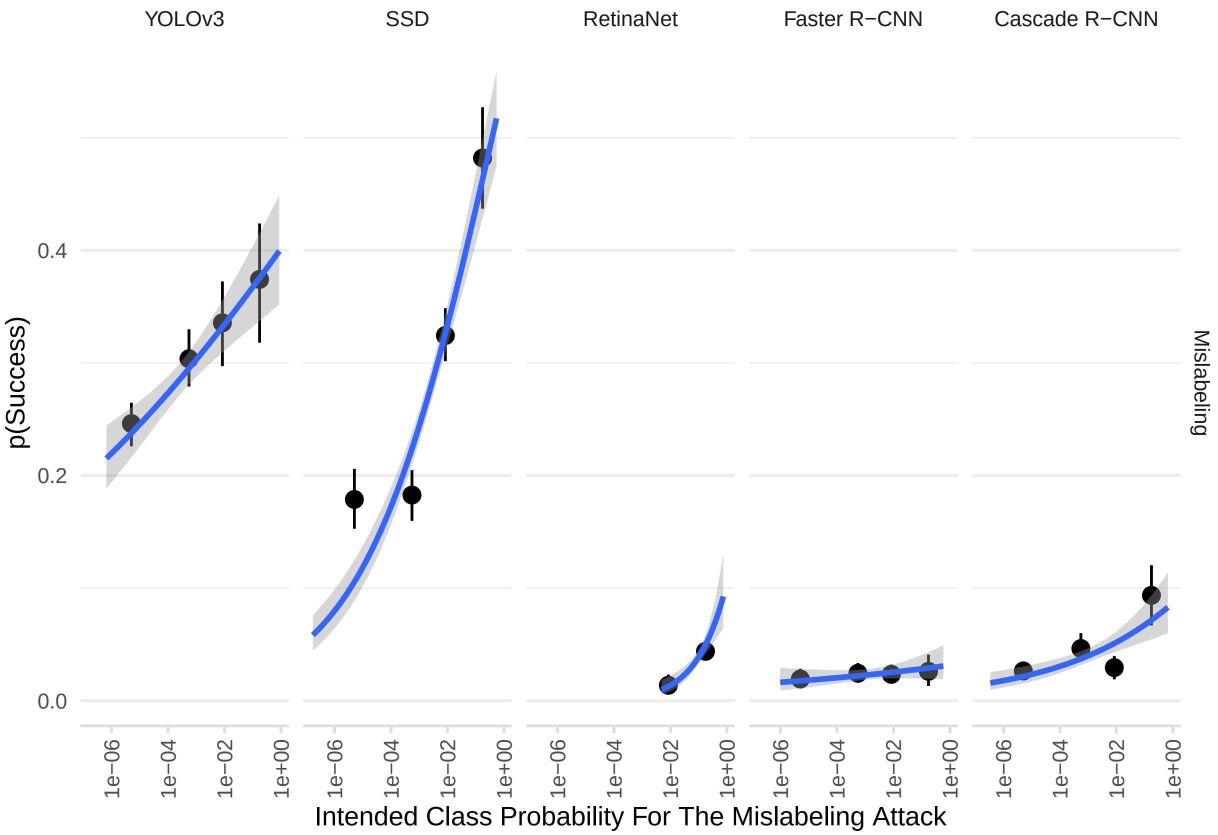


Figure 5: **Although intended class probability seem to increase success rates for the mislabeling attack, it does not predict success rates after controlling for target class confidence, except for RetinaNet (Table 7):** The binned summaries and regression trendlines graph success proportion against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

accuracy * confidence	-1.014	1.751	-0.579	0.562	-4.446	2.423

```
# 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
```

```
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
```

```
## Warning in bold_tex(str_to_sentence(main_pt), norm): NAs introduced by coercion
```

```
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.
```

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

```
## ----------log(sample_mislabel_proba)----------
## Total 5 predictors:
## 2 (40%) significant;
## 1 (20%) pos
```

```
## # 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 RetinaNet  Mislabeling log(samp~    0.683     0.325      2.10   0.036    0.036
## # i 1 more variable: conf.high <dbl>
```

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

```
## ----------log(sample_mislabel_proba):bbox_conf----------
## Total 5 predictors:
## 2 (40%) significant;
## 2 (40%) both
```

```
## # 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 YOLOv3     Mislabeling log(samp~    0.363     0.057      6.34   0          0.251
## 2 SSD        Mislabeling log(samp~    0.144     0.064      2.26   0.024      0.02
## # i 1 more variable: conf.high <dbl>
```

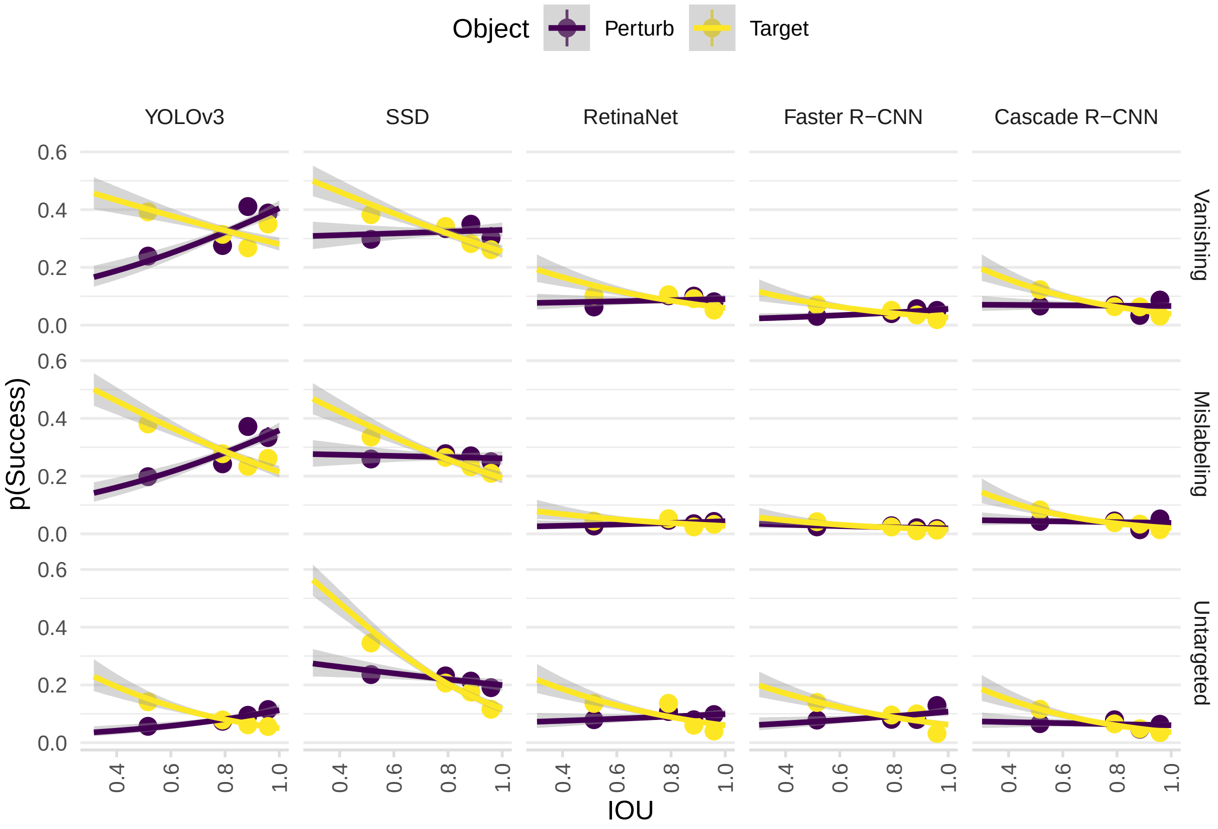
```r
print_statistics(reg_est, table_caption(glue("log({pred_name}) {att_name}, with predicted class's confi
```

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)	*	-0.202	0.040	-5.028	0.000	-0.281	-0.123
	confidence		0.758	0.485	1.563	0.118	-0.192	1.712
	log(probability) * confidence	*	0.363	0.057	6.337	0.000	0.251	0.476
SSD	log(probability)		0.058	0.047	1.242	0.214	-0.033	0.150
	confidence		-0.161	0.429	-0.375	0.707	-1.001	0.682
	log(probability) * confidence	*	0.144	0.064	2.264	0.024	0.020	0.270
RetinaNet	log(probability)	*	0.683	0.325	2.101	0.036	0.036	1.308
	confidence	*	-8.137	1.846	-4.408	0.000	-11.802	-4.567
	log(probability) * confidence		-0.842	0.703	-1.198	0.231	-2.183	0.571
Faster R-CNN	log(probability)		0.018	0.115	0.156	0.876	-0.209	0.242
	confidence	*	-5.405	1.292	-4.183	0.000	-7.955	-2.880
	log(probability) * confidence		-0.165	0.167	-0.987	0.324	-0.489	0.167
Cascade R-CNN	log(probability)		-0.022	0.095	-0.237	0.813	-0.210	0.162
	confidence		-1.592	0.871	-1.827	0.068	-3.282	0.139
	log(probability) * confidence		0.094	0.124	0.756	0.450	-0.146	0.340

```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")
```

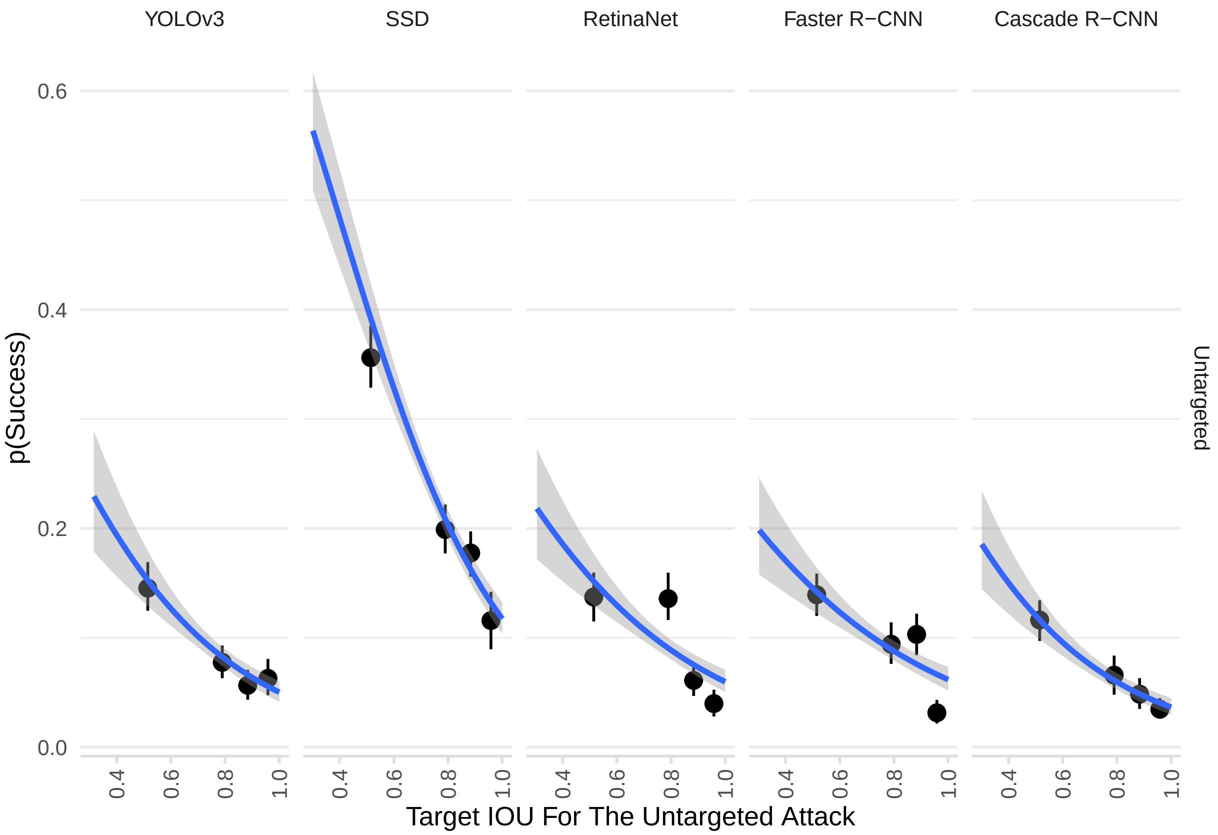


Figure 6: **Target IOU for the untargeted attack increases success rates on all models:** The binned summaries and regression trendlines graph success proportion against target IOU for the untargeted attack in the randomized attack experiment. Bins are split into quantiles. Errors are 95% confidence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

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

```
## Warning in bold_tex(str_to_sentence(main_pt), norm): NAs introduced by coercion
```

```
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_~    -2.53     0.341     -7.42       0    -3.19
## 2 SSD           Untargeted  bbox_~    -3.25     0.235    -13.8        0    -3.72
## 3 RetinaNet     Untargeted  bbox_~    -2.13     0.308     -6.90       0    -2.73
## 4 Faster R-CNN  Untargeted  bbox_~    -1.90     0.294     -6.46       0    -2.47
## 5 Cascade R-CNN Untargeted  bbox_~    -2.57     0.318     -8.06       0    -3.19
## # 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	*	-2.526	0.341	-7.417	0	-3.189	-1.853
SSD	bbox_iou_eval	*	-3.254	0.235	-13.838	0	-3.716	-2.794
RetinaNet	bbox_iou_eval	*	-2.130	0.308	-6.904	0	-2.730	-1.520
Faster R-CNN	bbox_iou_eval	*	-1.899	0.294	-6.460	0	-2.471	-1.318
Cascade R-CNN	bbox_iou_eval	*	-2.566	0.318	-8.062	0	-3.187	-1.938